

# **COMPTE RENDU DU TL**

Nathan Lewy, Pierre Sainctavit, Timo Descazeaud

# **Partie 1 :**

# **Banks de filtre**

# Décomposition d'une somme de sinus

- Signal harmonique composé d'une somme de 4 sinus
- $f_0 = 880 \text{ Hz}$
- $f_e = 8000 \text{ Hz}$

$$s(t) = \sum_{h=1}^4 \sin(2\pi h f_0 t)$$

- **Objectif :** effectuer des décompositions successives afin de n'avoir qu'un sinus par bande, et retrouver les fréquences de ceux-ci

# Décomposition d'une somme de sinus

- Première décomposition ( $f_e \rightarrow f_{e1} = 4000$  Hz)
  - Basses fréquences :  $f_0 = 880$  Hz et  $2f_0 = 1760$  Hz inchangés
  - Hautes fréquences :  $-3f_0 = -2640$  Hz  $\rightarrow f_3 = 1360$  Hz  
 $-4f_0 = -3520$  Hz  $\rightarrow f_4 = 480$  Hz

# Décomposition d'une somme de sinus

- Deuxième décomposition ( $f_e \rightarrow f_{e2} = 2000$  Hz)
  - BF de BF :  $f_0 = 880$  Hz inchangé
  - HF de BF :  $-2f_0 = -1760$  Hz  $\rightarrow f_2 = 240$  Hz
  - BF de HF :  $f_4 = 480$  Hz inchangé
  - HF de HF :  $-f_3 = -1360$  Hz  $\rightarrow f_{3*} = 640$  Hz

# Décomposition d'une somme de sinus

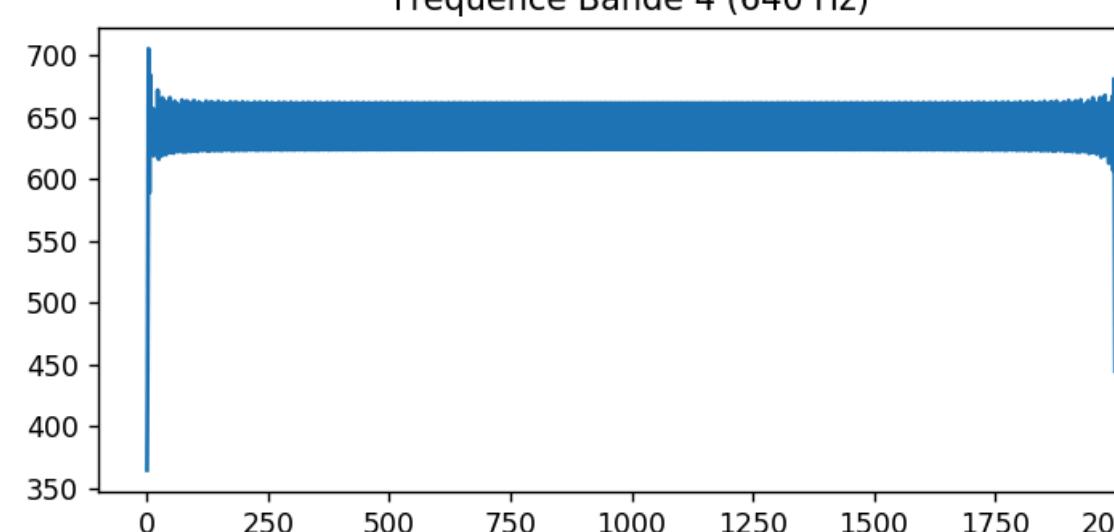
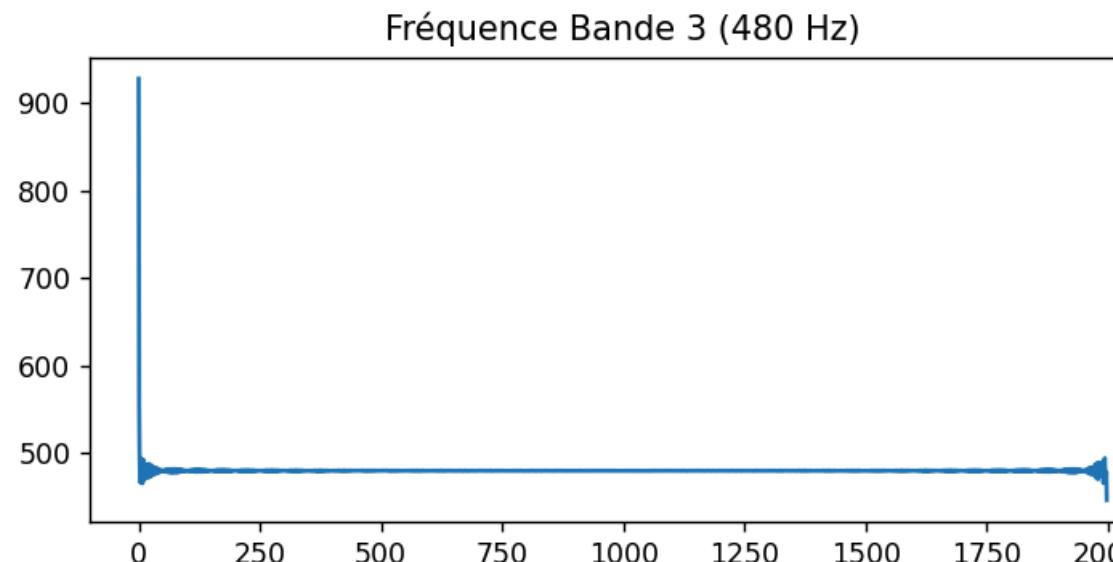
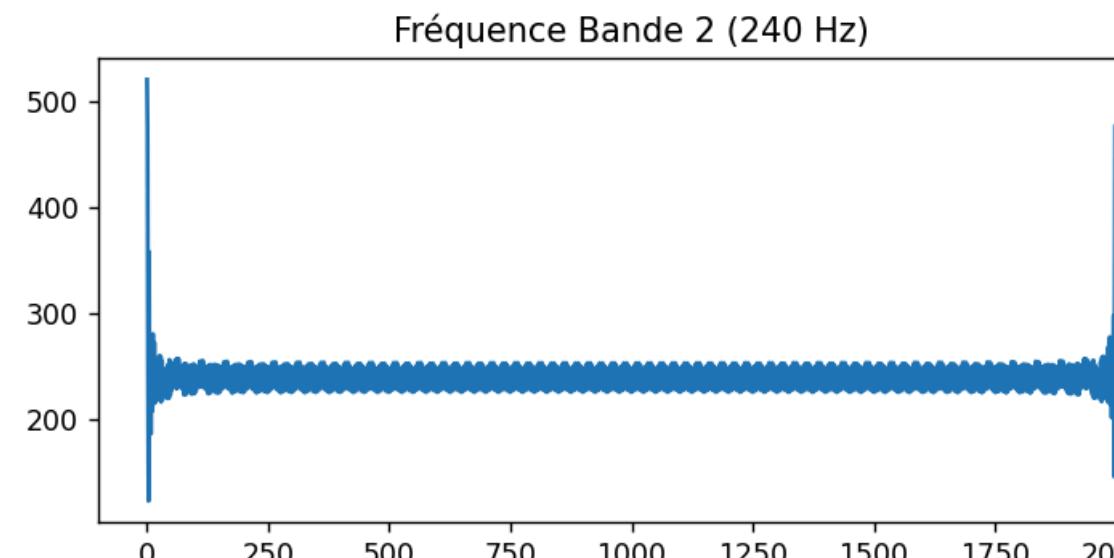
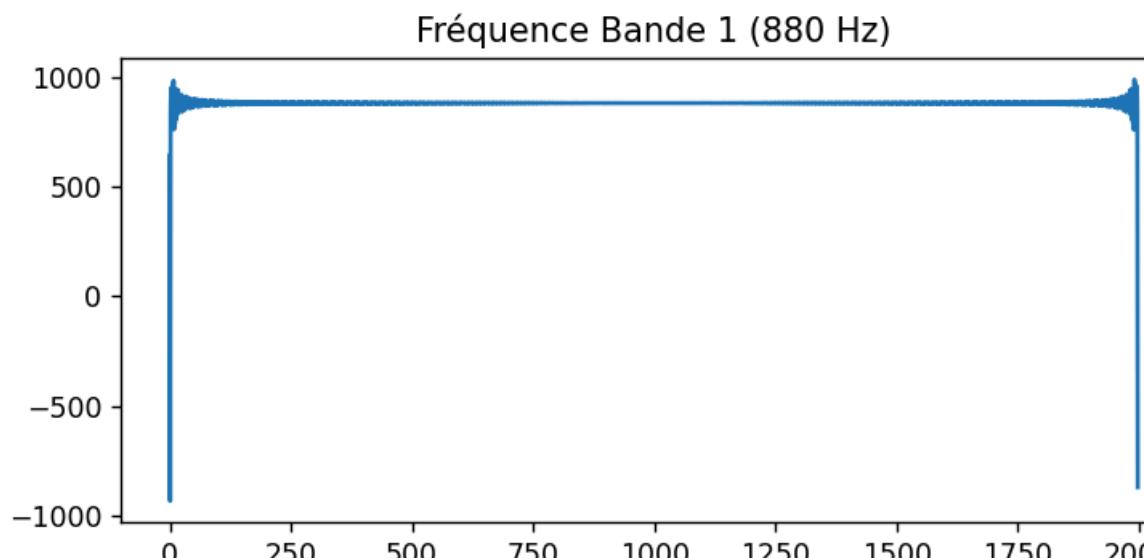
**Conclusion :**

- $f_0 = 880 \text{ Hz} \rightarrow 880 \text{ Hz}$
- $2f_0 = 1760 \text{ Hz} \rightarrow 240 \text{ Hz}$
- $3f_0 = 2640 \text{ Hz} \rightarrow 640 \text{ Hz}$
- $4f_0 = 3520 \text{ Hz} \rightarrow 480 \text{ Hz}$

# Décomposition d'une somme de sinus

**Application :**

Deux décompositions + filtre de Hilbert



Bande 1 : 880.00 Hz  
Bande 2 : 241.60 Hz  
Bande 3 : 480.01 Hz  
Bande 4 : 645.37 Hz

Extraction de la médiane

# Ajout d'un vibrato et spectrogramme

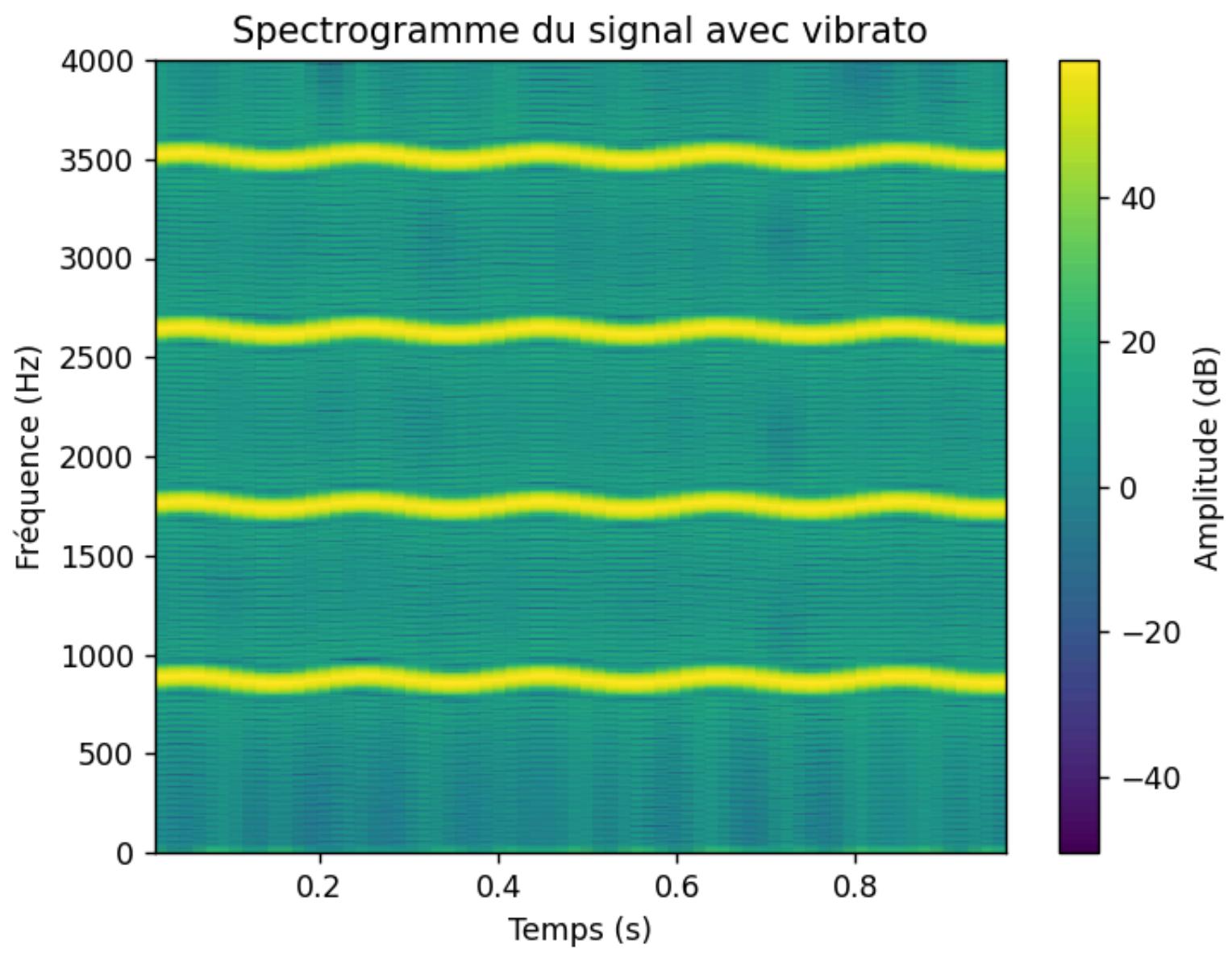
- Ajout d'un vibrato tel que  $A_{\text{vibrato}} = 20 \text{ Hz}$  et  $f_{\text{vibrato}} = 5 \text{ Hz}$

$$f(t) = f_0 + A_v \sin(2\pi f_v t)$$

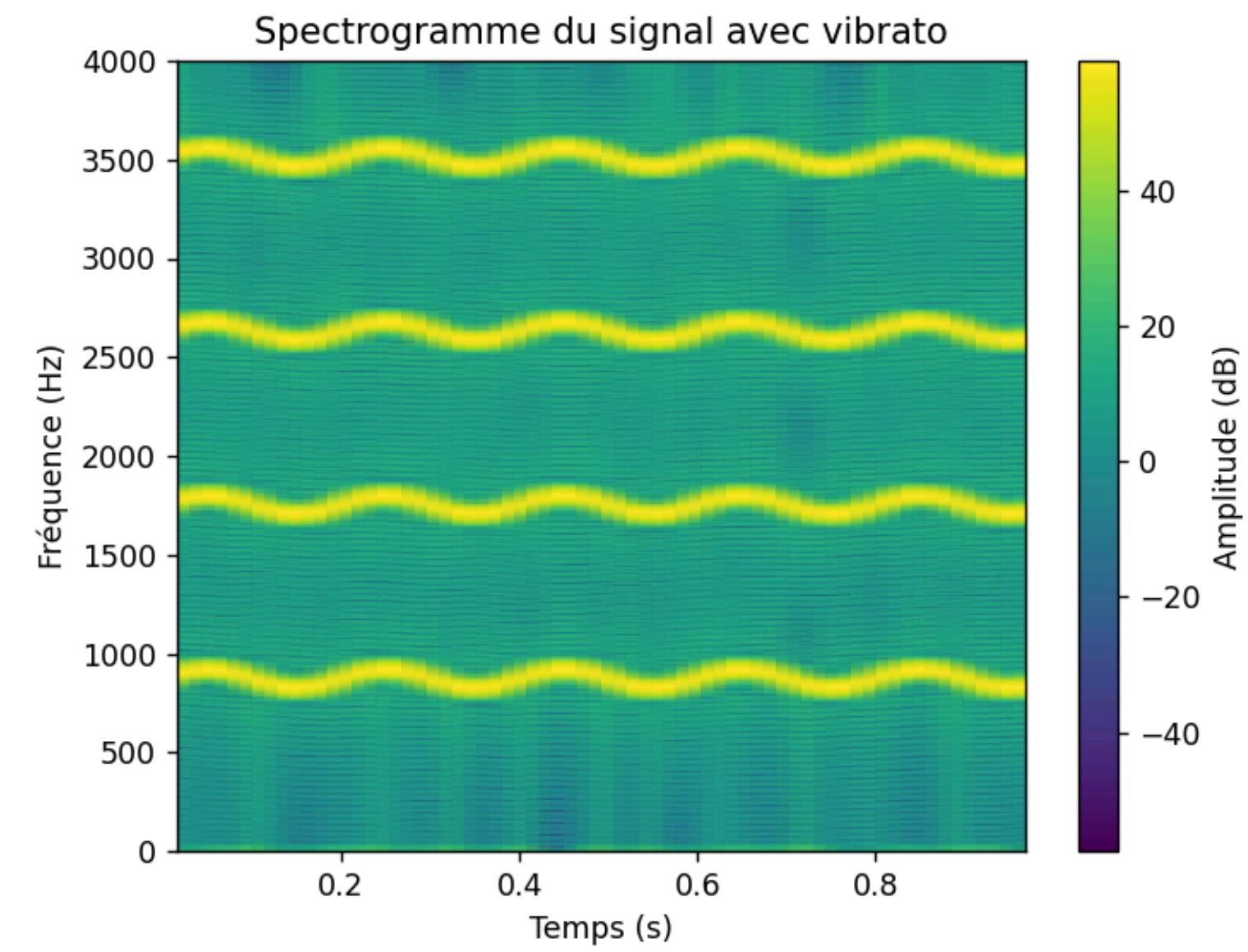
$$s_v(t) = \sum_{h=1}^4 \sin(2\pi h f_0 t - \frac{A_v}{f_v} \cos(2\pi f_v t))$$

# Ajout d'un vibrato et spectrogramme

$A_v = 20 \text{ Hz}$

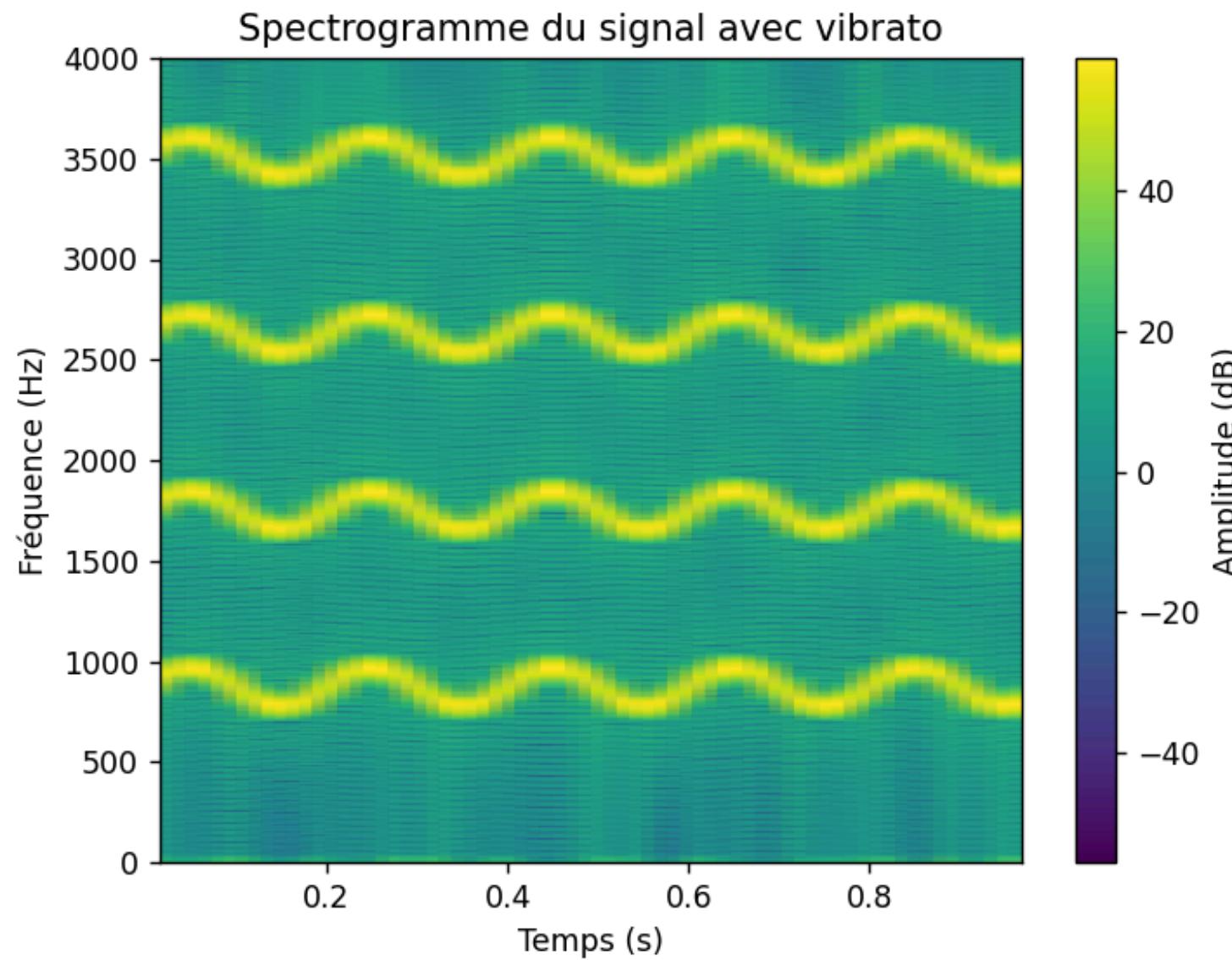


$A_v = 50 \text{ Hz}$

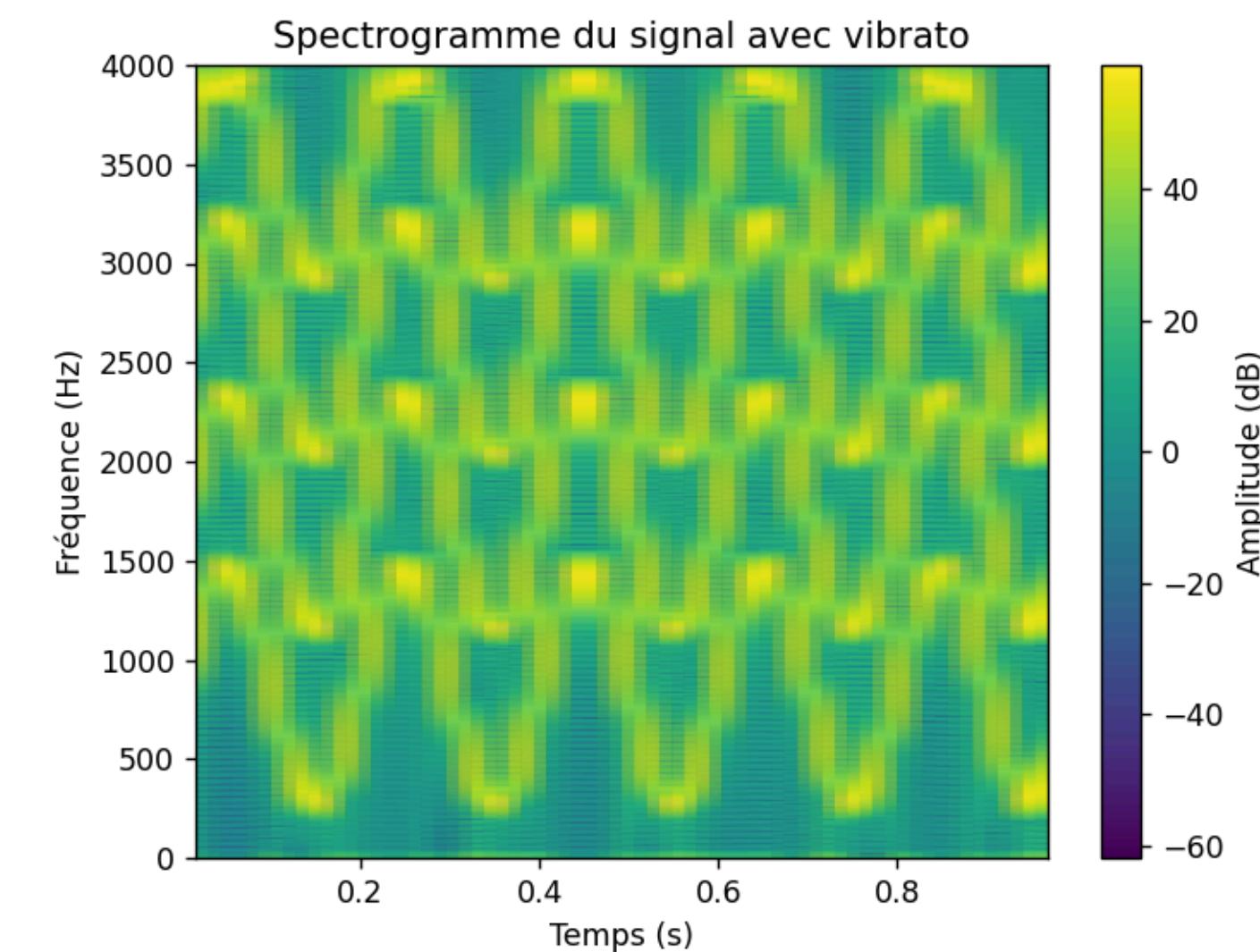


# Ajout d'un vibrato et spectrogramme

$A_v = 100 \text{ Hz}$



$A_v = 600 \text{ Hz}$



# **Partie 2 :**

# **Ondelettes**

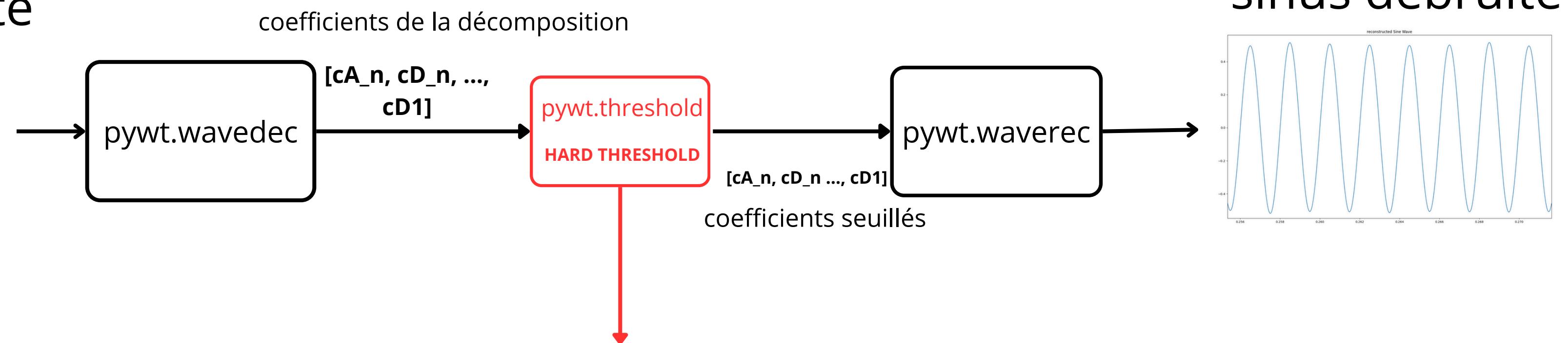
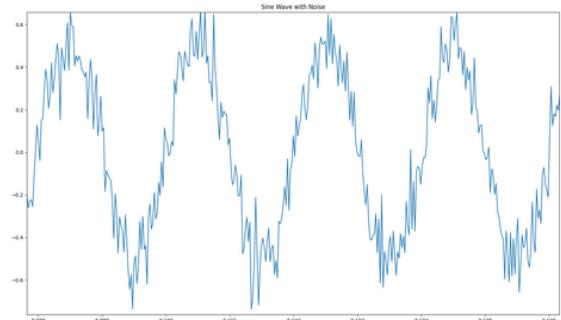
# Débruitage

- > Signal sinusoïdal de 500 Hz échantilloné à 44100 Hz,  
durée 1 seconde
- > Signal noyé dans du bruit blanc
- > **Objectif** : obtenir un signal sans bruit

# Débruitage

--> Comment faire ? :

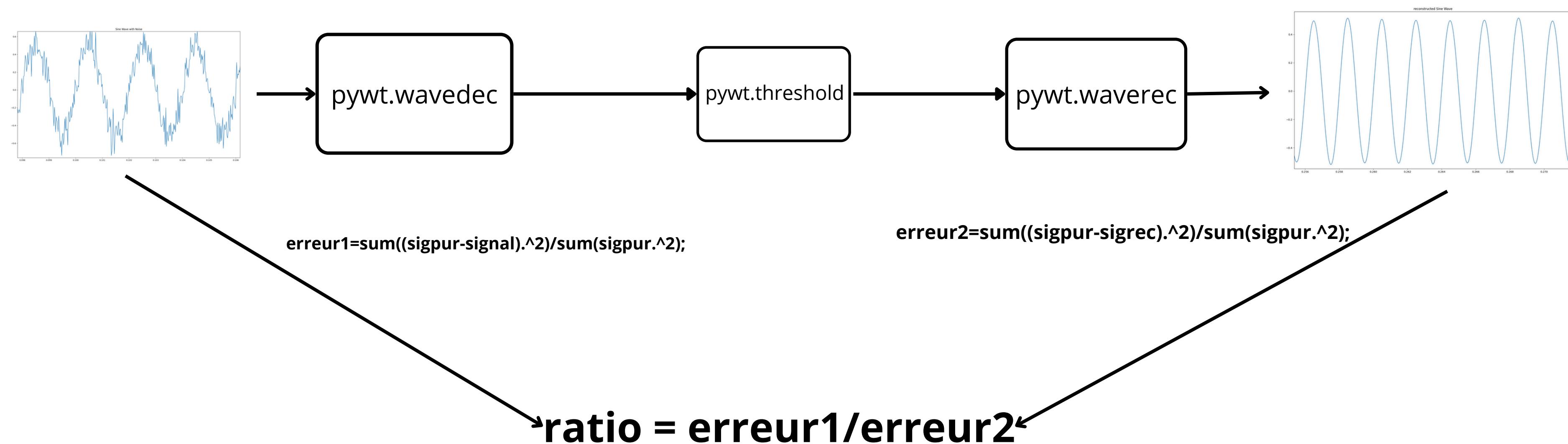
sinus bruité



--> On suppose que le bruit s'est réparti uniformément sur tout les niveau de décompositions et on ne garde donc que les coefficients supérieur à un certain seuil

# Débruitage

--> Est-ce que ça a marché ? :



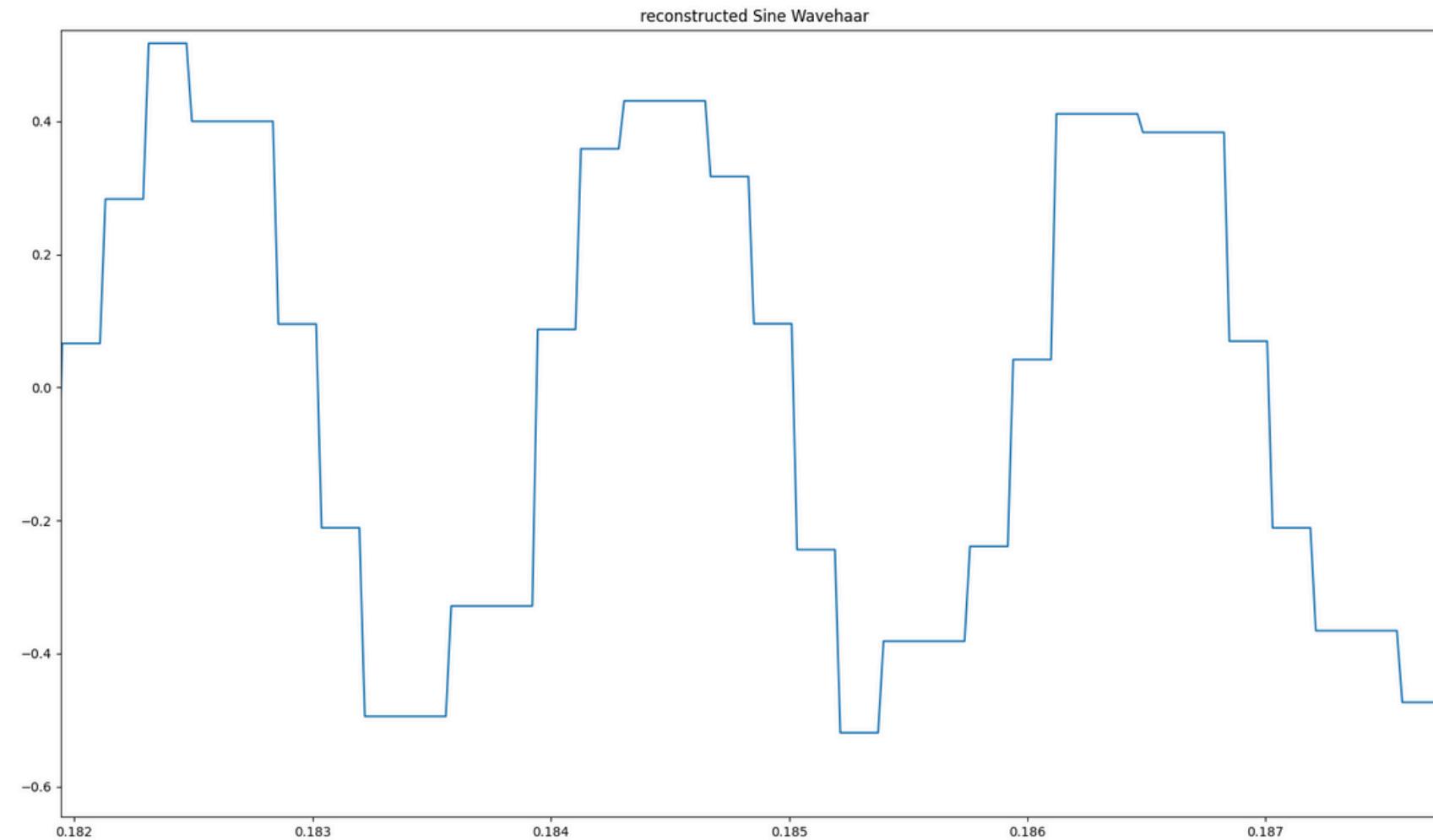
# Débruitage

--> Résultats :

--> Essais avec différentes ondelettes

--> 30 niveaux de décompositions, amplitude bruit = 0.1, amplitude sinus = 0.5, seuil = 0.4, 20 répétitions de l'expérience

**Haar**



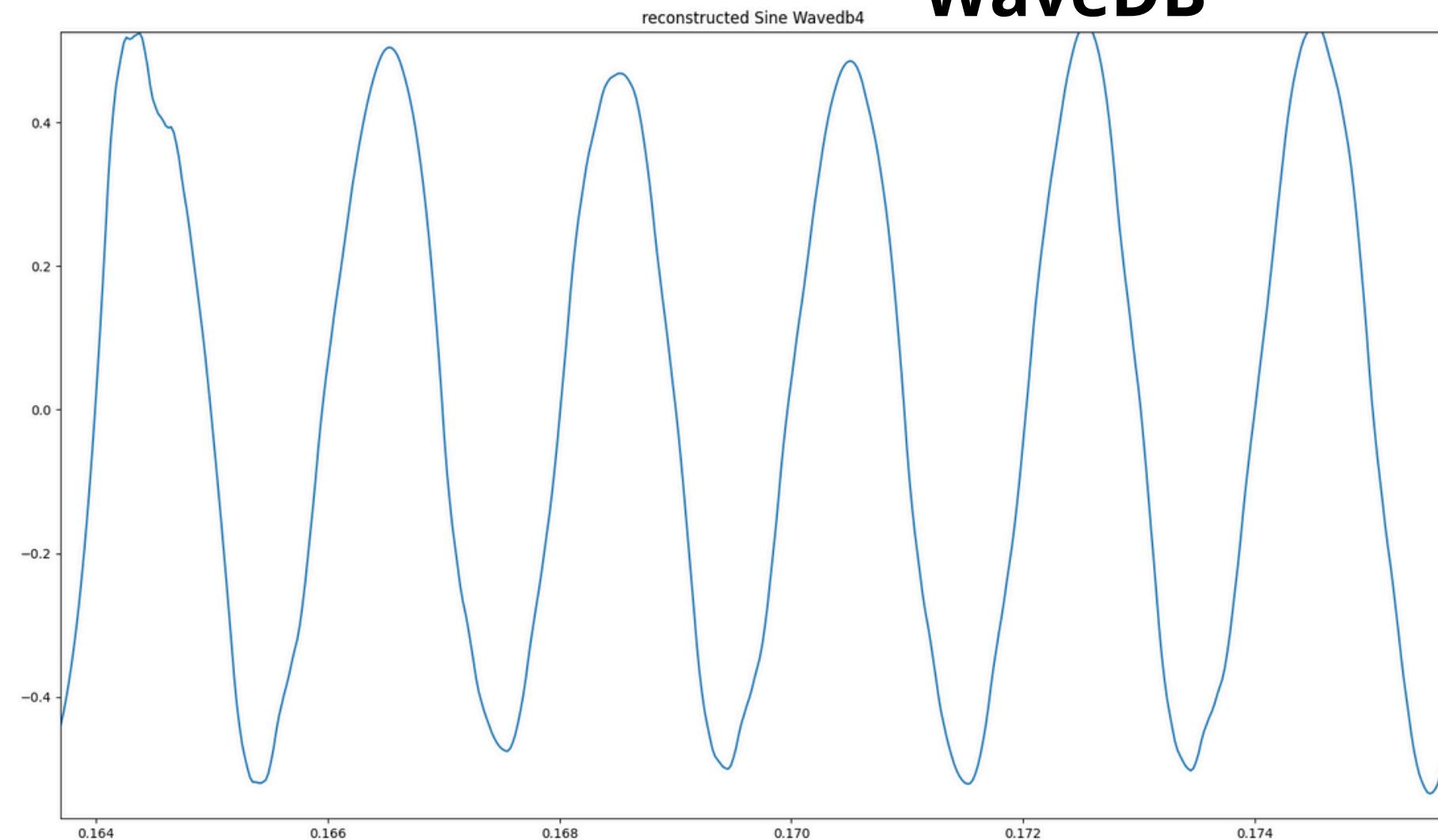
moyenne du SNR : 1.3353084271357458  
écart type du SNR : 2.220446049250313e-16

# Débruitage

--> Résultats :

--> Essais avec différentes ondelettes

**WaveDB**

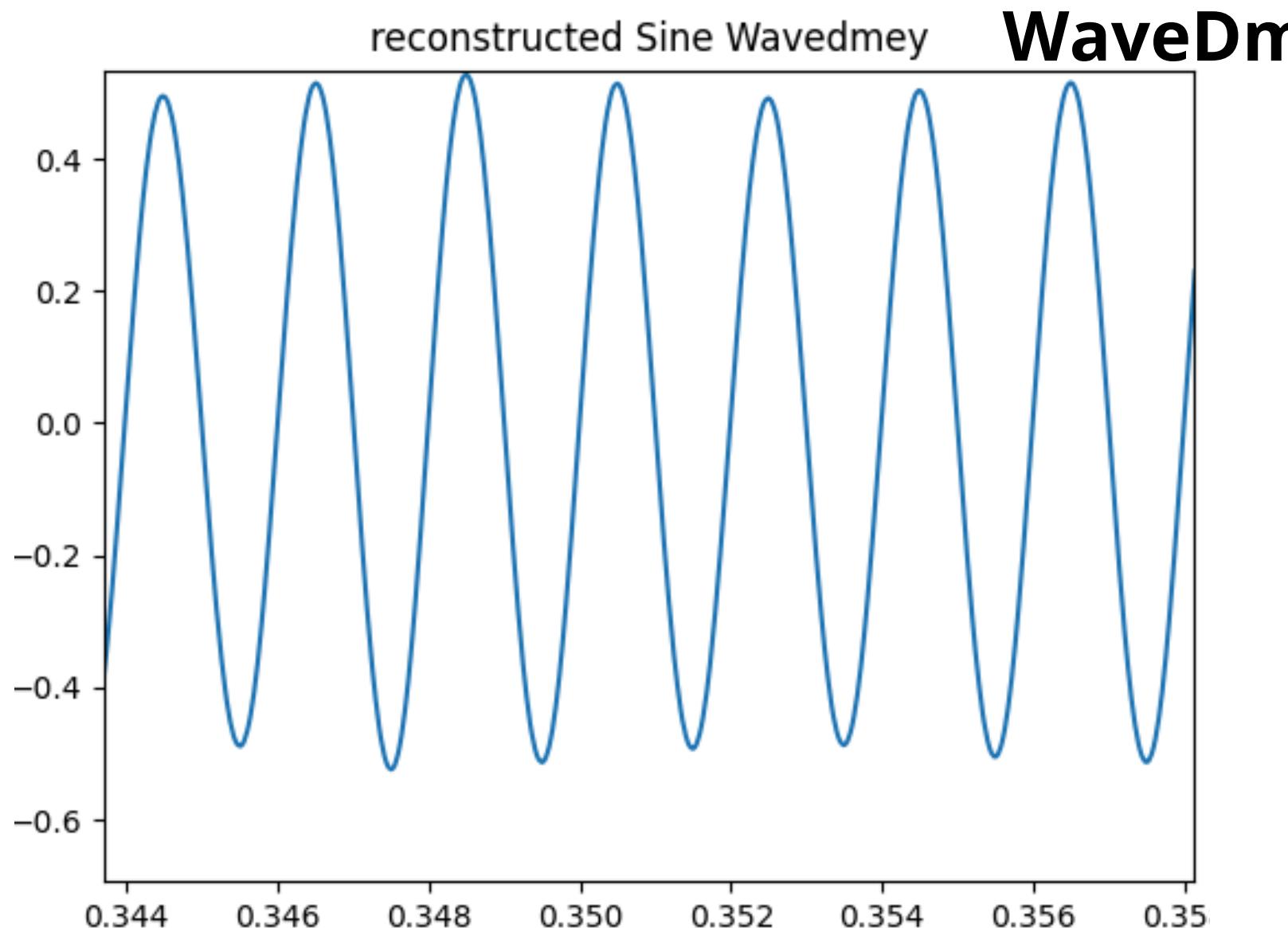


moyenne du SNR : 5.9971595396879165  
écart type du SNR : 8.881784197001252e-16

# Débruitage

--> Résultats :

--> Essais avec différentes ondelettes



moyenne du SNR : 33.93587735403002  
écart type du SNR : 0.0

--> Dmey très clairement meilleure

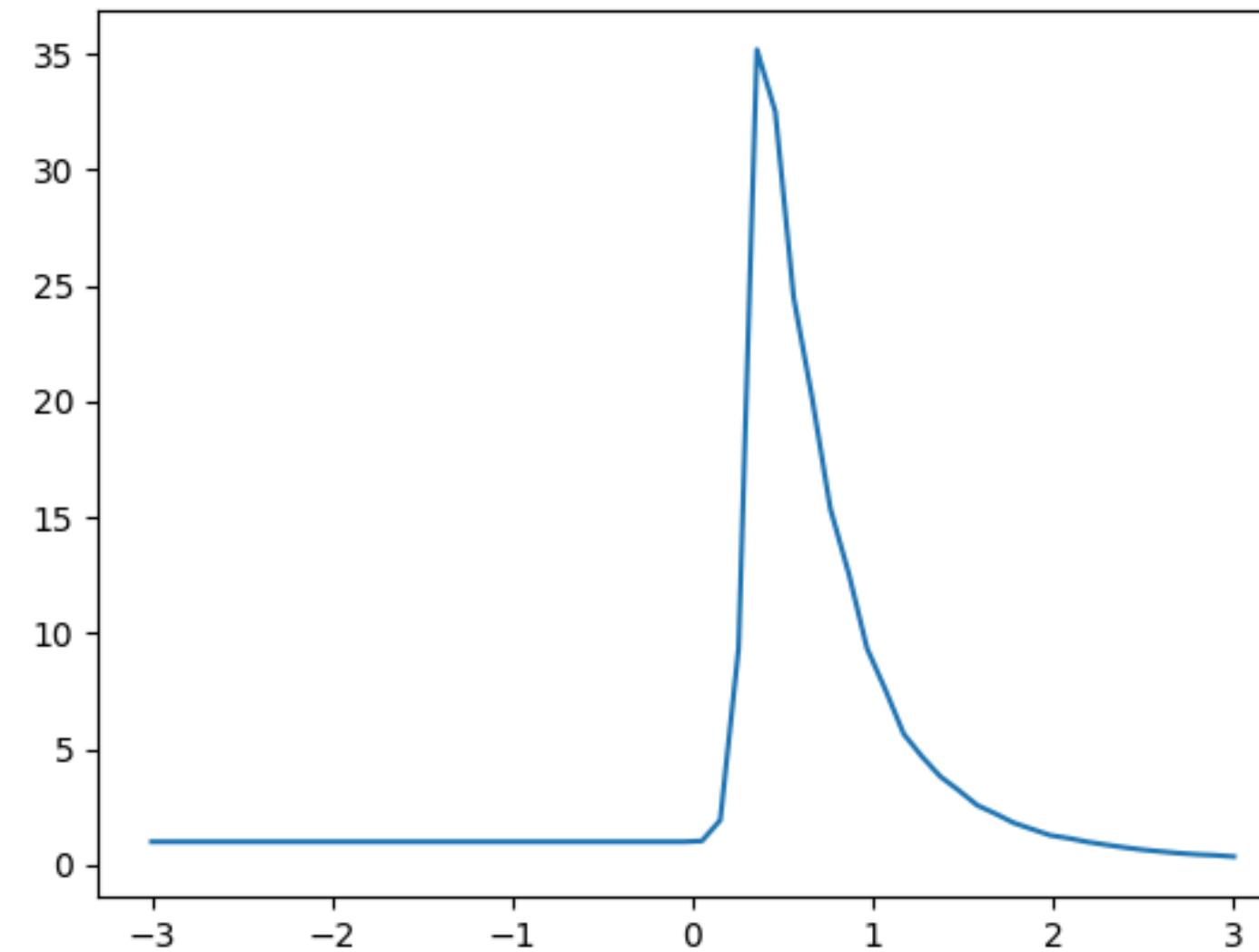
# Débruitage

--> Résultats :

--> Essais avec différents seuils

--> 30 niveaux de décompositions, amplitude bruit = 0.1, amplitude sinus = 0.5, 20 répétitions de l'expérience,

SNR ratio for different thresholds, dmeye wavelet



**DMEY**

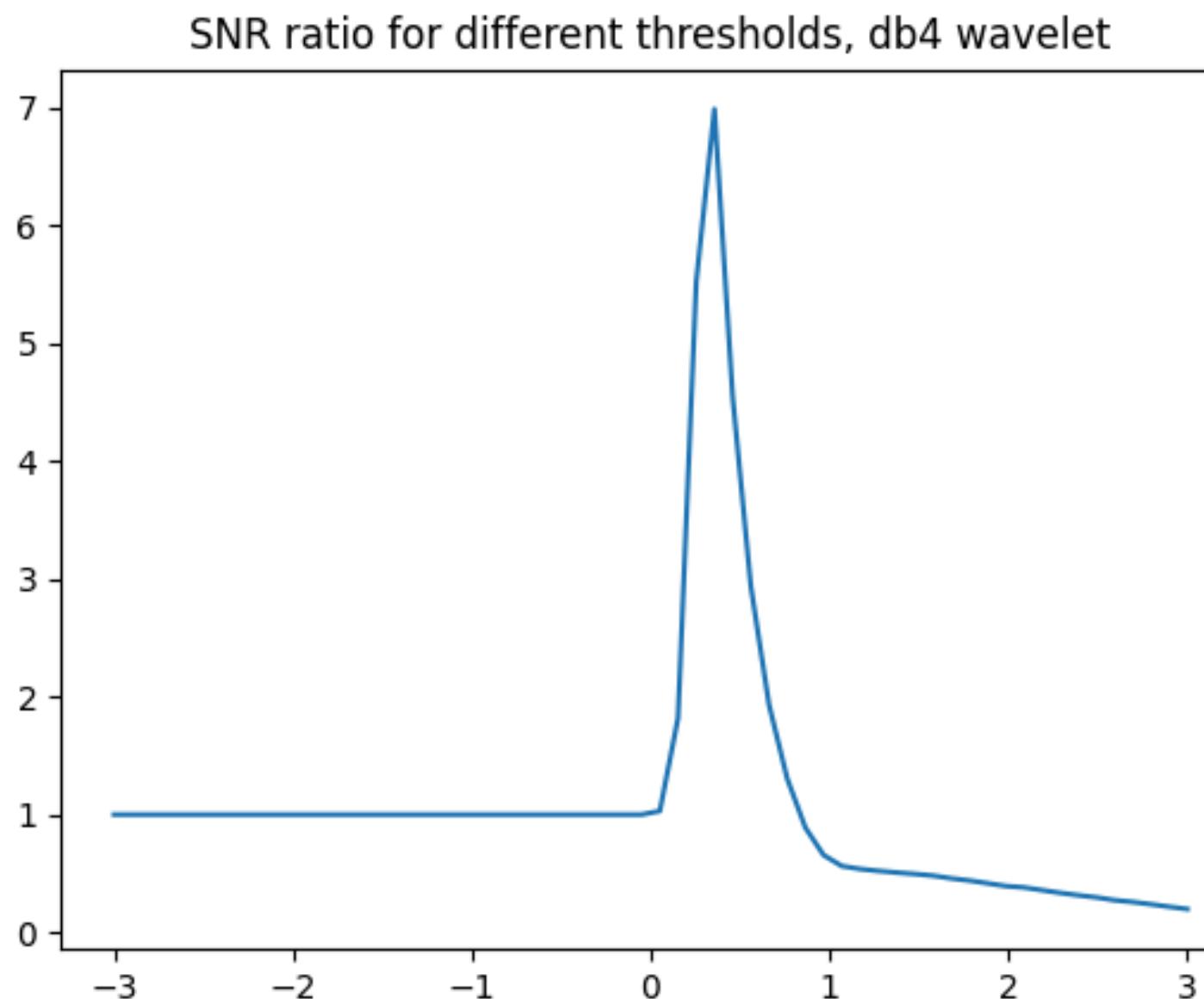
--> Environ 1 seconde de temps de calcul

# Débruitage

--> Résultats :

--> Essais avec différents seuils

--> 30 niveaux de décompositions, amplitude bruit = 0.1, amplitude sinus = 0.5, 20 répétitions de l'expérience,



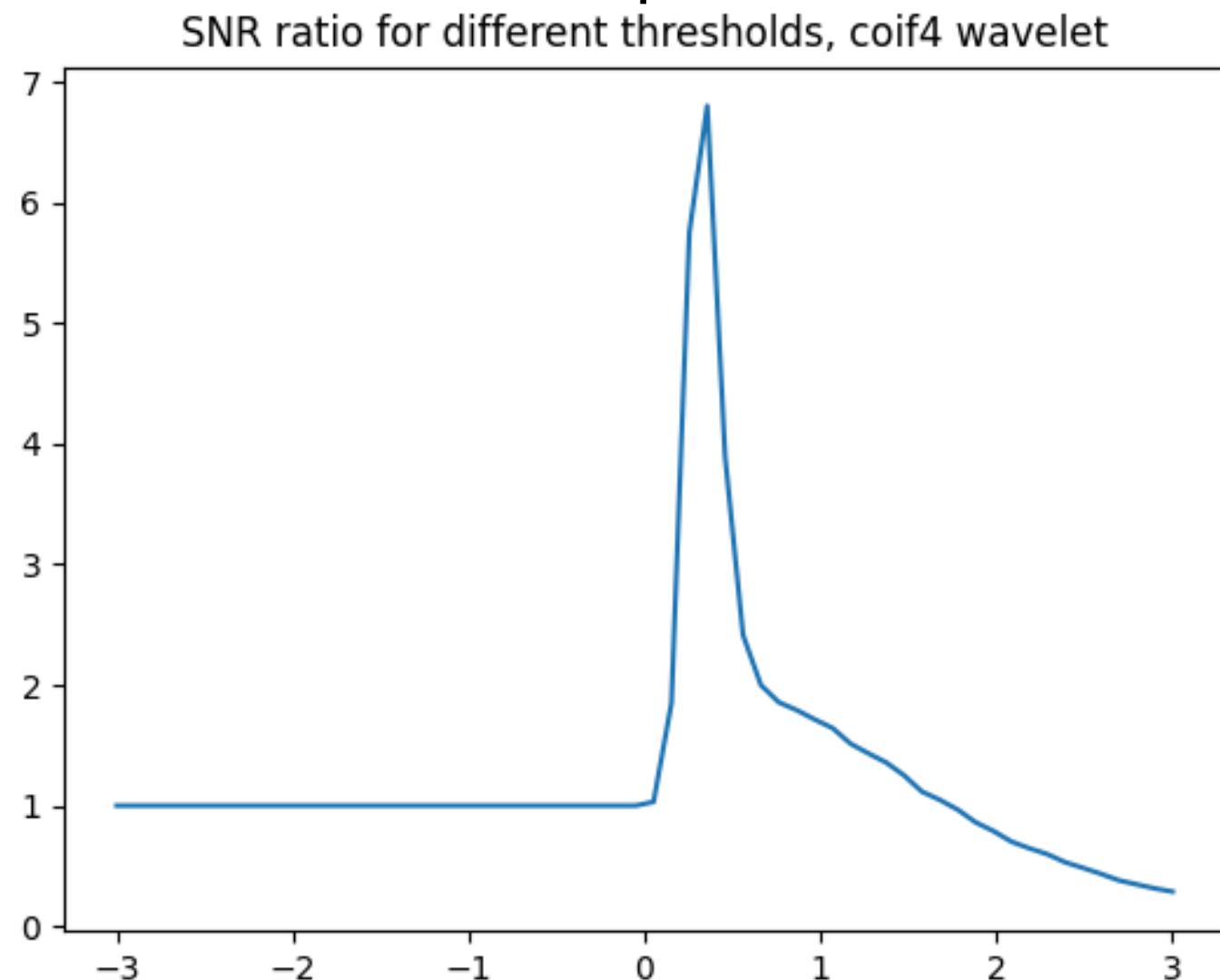
**DB4**

# Débruitage

--> Résultats :

--> Essais avec différents seuils

--> 30 niveaux de décompositions, amplitude bruit = 0.1, amplitude sinus = 0.5, 20 répétitions  
de l'expérience,



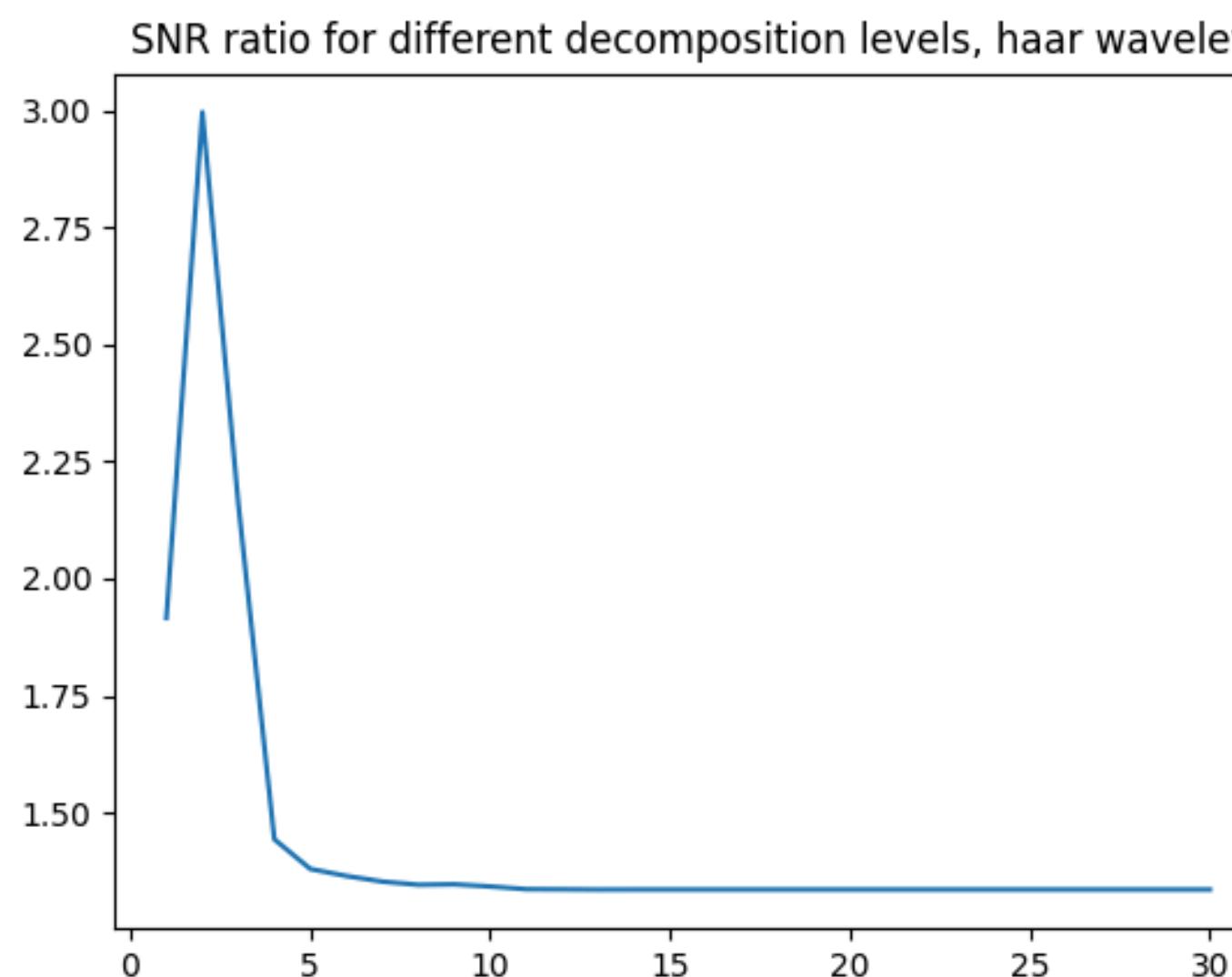
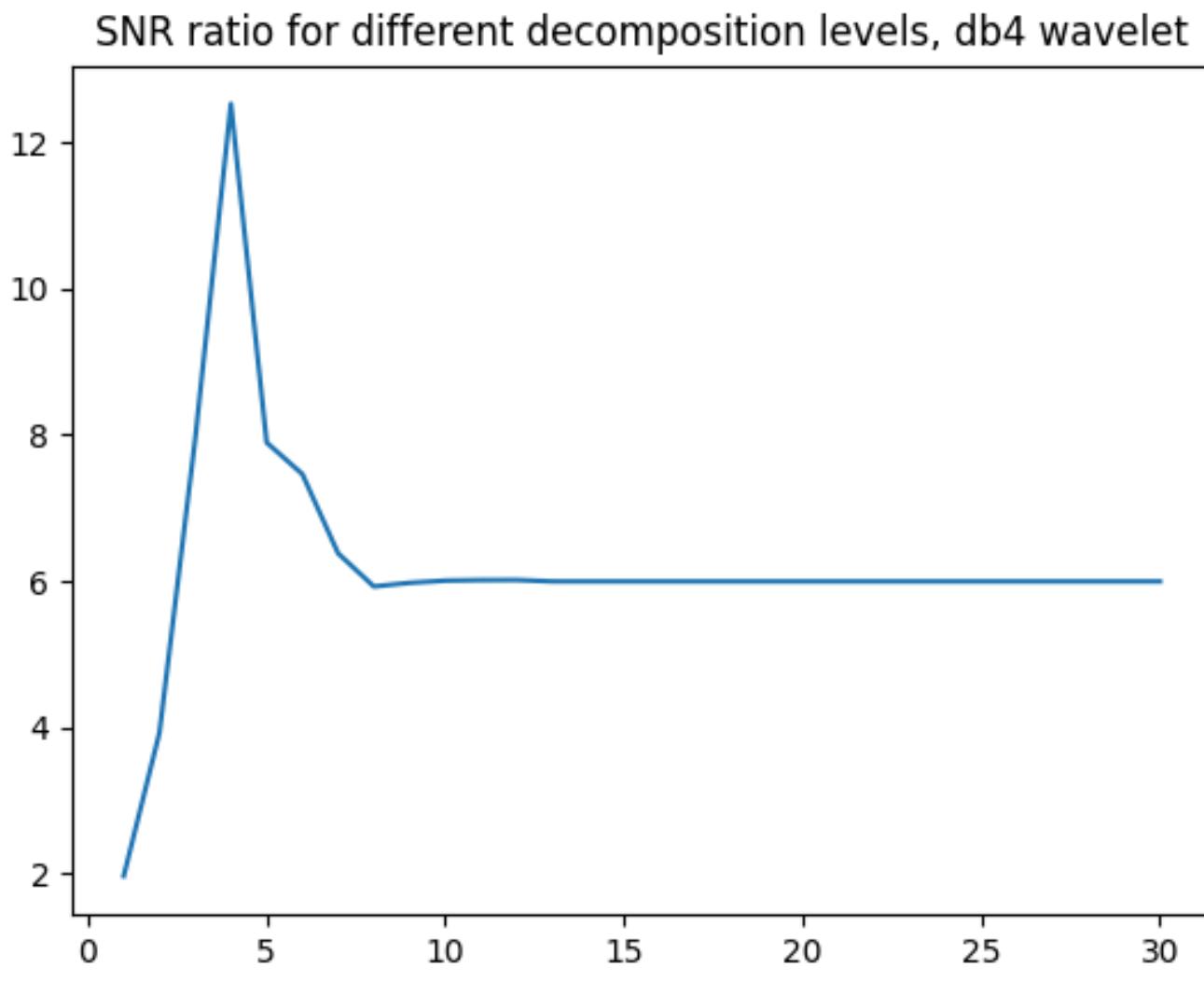
**COIF4**

--> dmey surpassé clairement les autres

# Débruitage

--> Résultats :

--> Essais avec différents niveau de décomposition



# Débruitage

--> Résultats :

--> Essais en faisant varier tout les paramètres

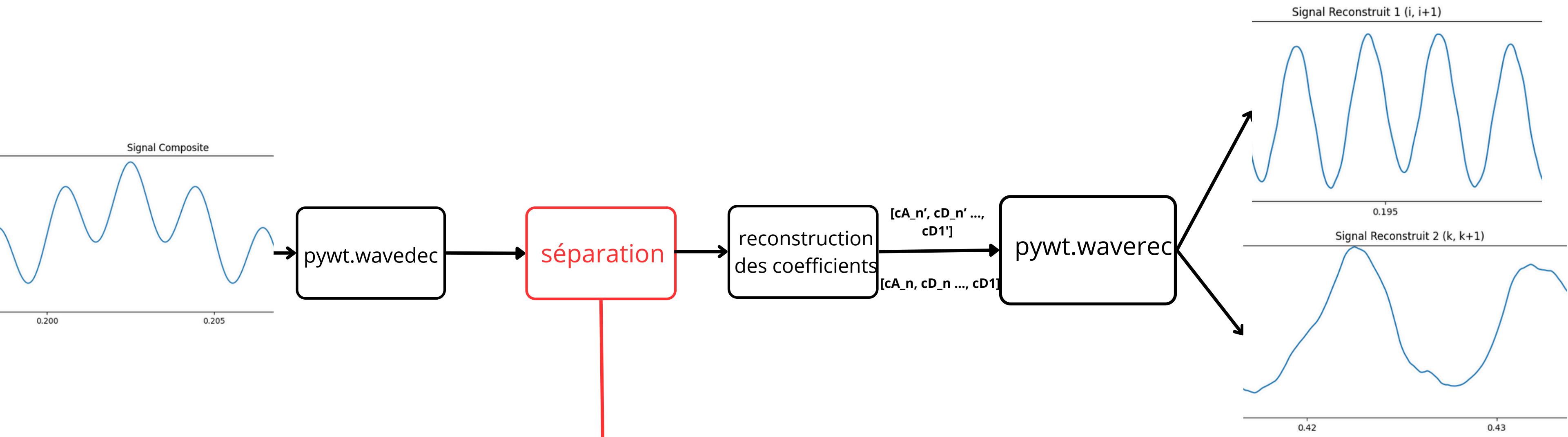
```
Best SNR Ratio: 37.26465892511551
Achieved with Wavelet: coif17
Achieved with Threshold: 0.4406779661016949
Achieved with level: 10
```

# Séparation

- > Somme de deux signaux : sinus à 500 Hz et sinus à 100 Hz
- > **Objectif** : séparer les signaux

# Séparation

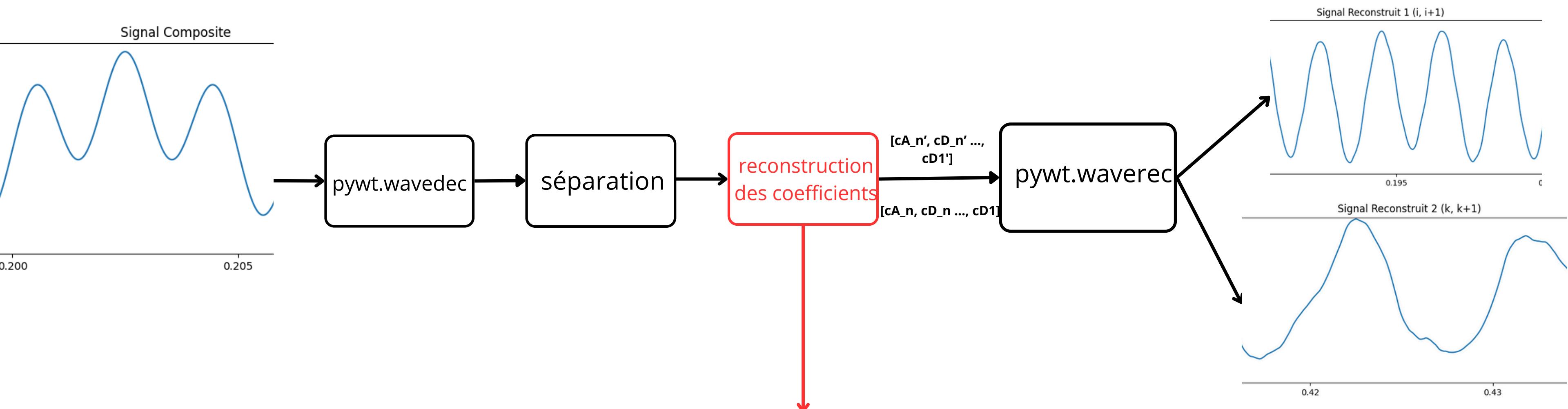
→ Comment faire ? :



- On choisit les quatres coefficients de détails les plus élevés au sens de la moyenne des valeurs absolues
- Ces coefficients ont pour indices  $(i, i+1)$ ,  $(k, k+1)$

# Séparation

→ Comment faire ? :



- On construit deux listes de coefficients de détails, égales à zéro
- On remplace dans la première liste les deux lignes qui ont pour indices ( $i, i + 1$ ) par les coefficients trouvés.  
Même chose pour la deuxième liste

# Séparation

--> Comment savoir si ça a marché ? :

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \tilde{y}_i)^2$$

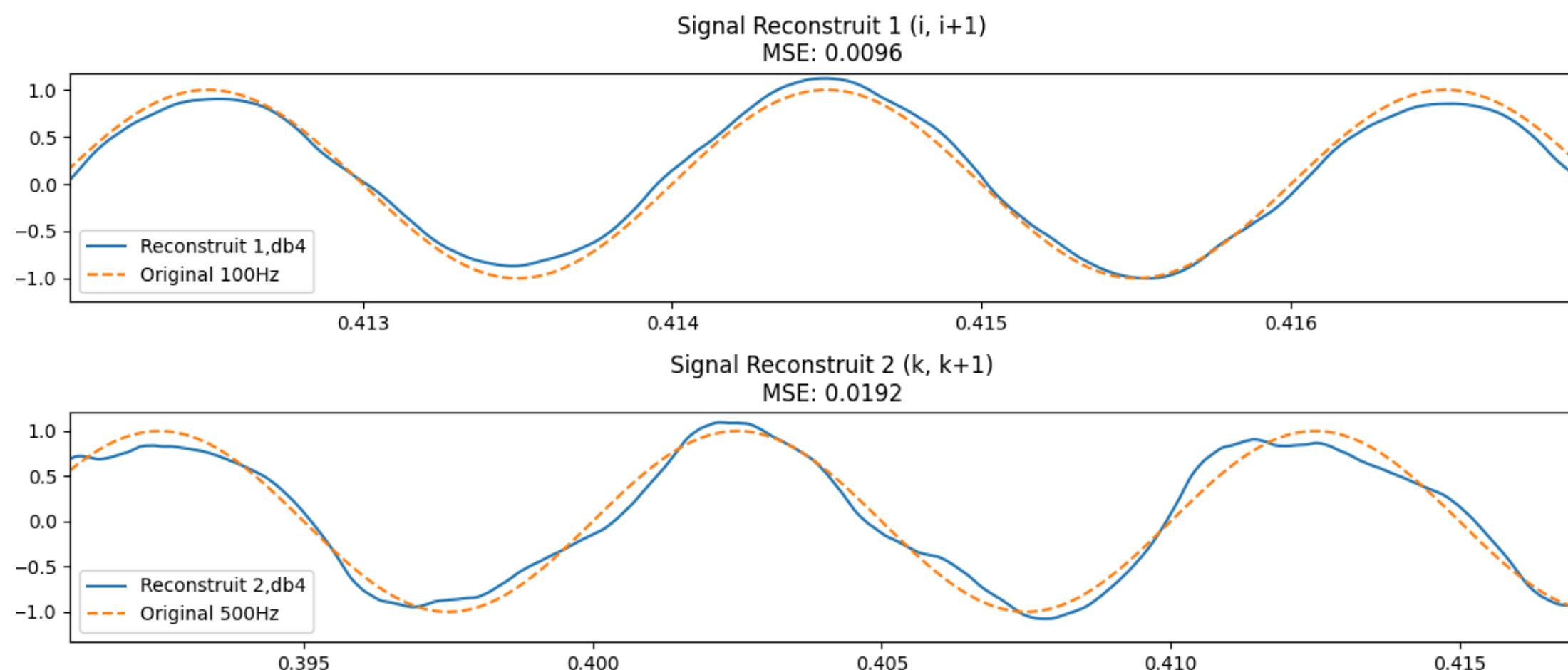
→ Où  $y_i$  est la sinus de base et  $y_i\text{ estimée}$  est la sinus reconstruite

# Séparation

## → Résultats

→  $f_{échantillonage} = 44100 \text{ Hz}$ , durée 1 seconde, niveau de la décomposition automatique

DB4, MSE = 0,0096, 0.0192

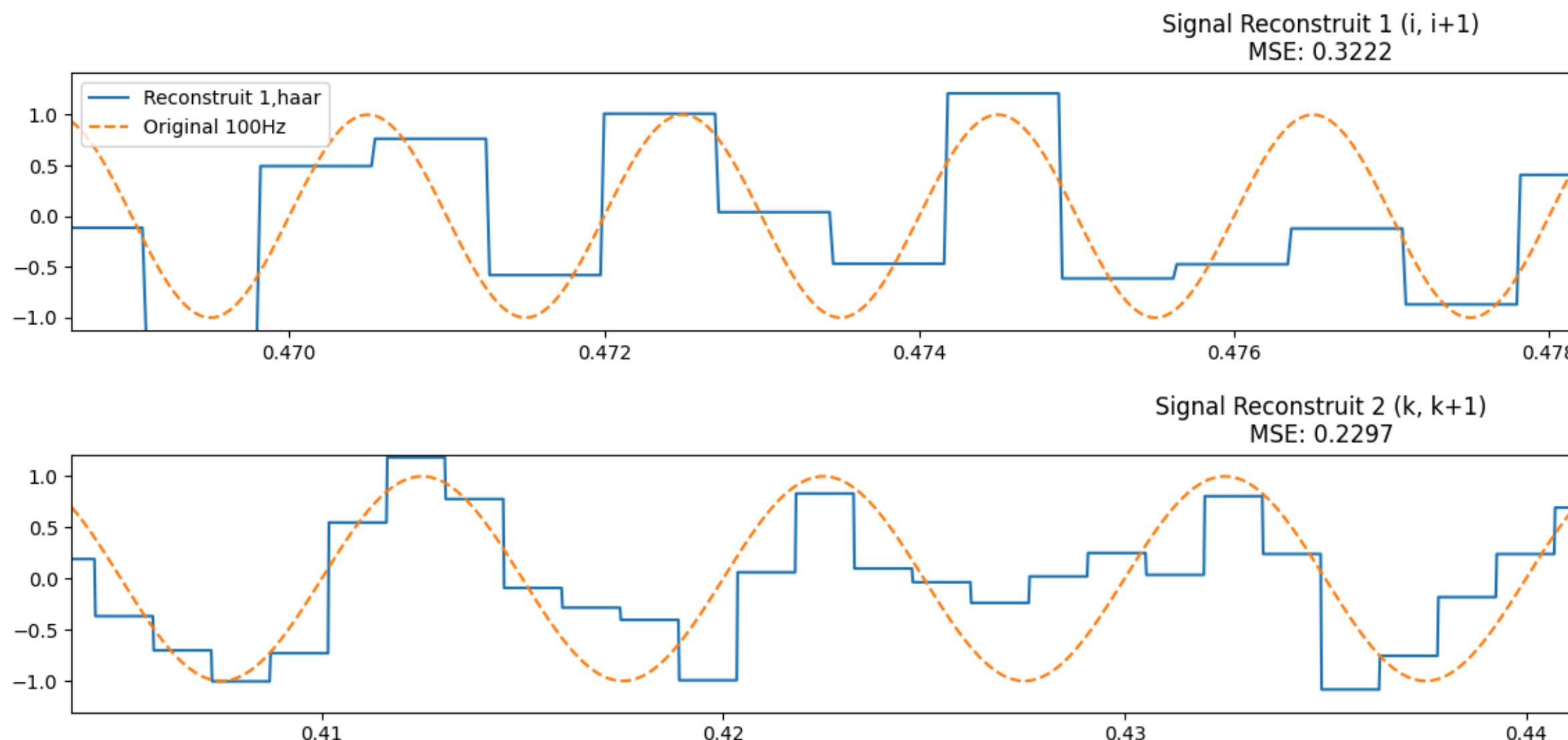


# Séparation

## → Résultats

→  $f_{\text{échantillonage}} = 44100 \text{ Hz}$ , durée 1 seconde, niveau de la décomposition automatique

HAAR, MSE = 0,322, 0.2297

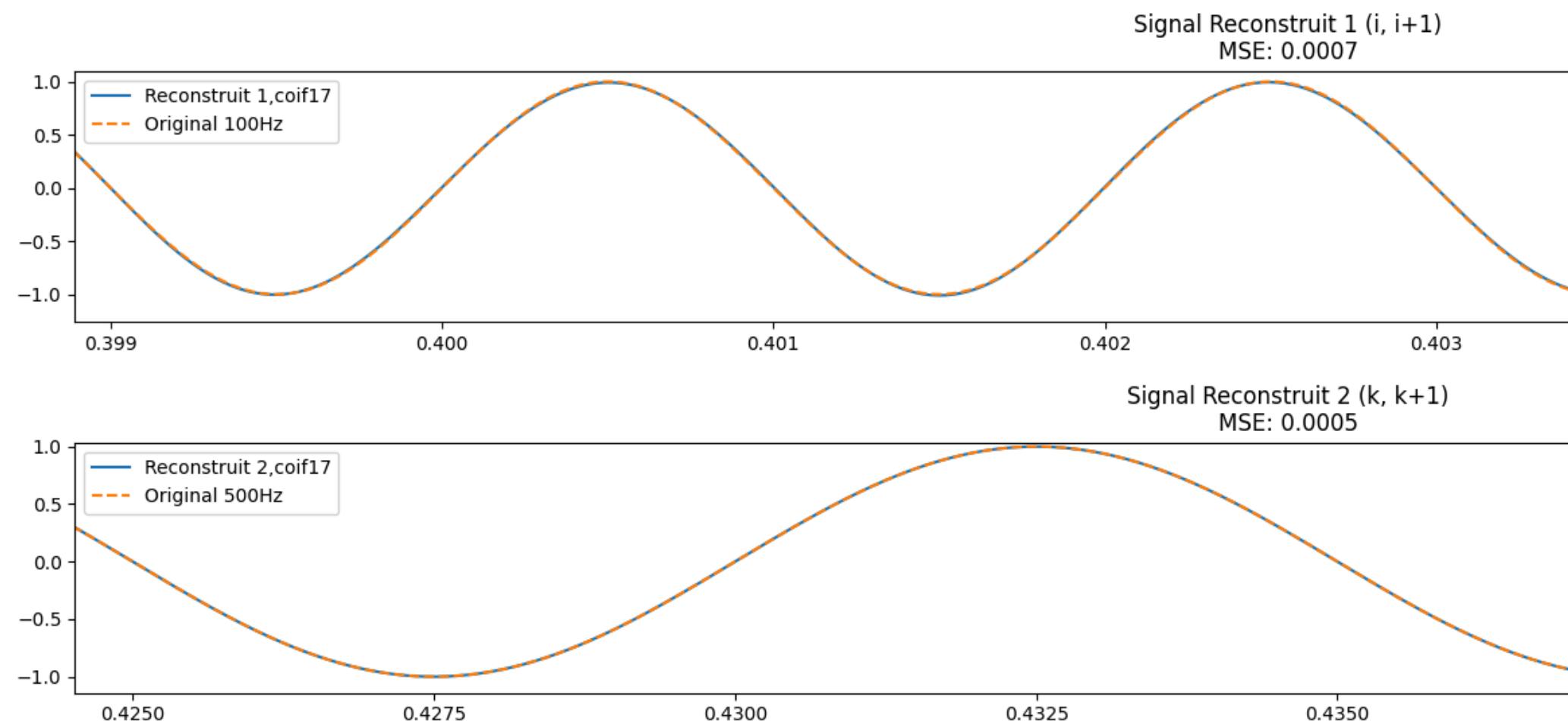


# Séparation

## → Résultats

→  $f_{échantillonage} = 44100 \text{ Hz}$ , durée 1 seconde, niveau de la décomposition automatique

COIF, MSE = 0,0007,  
0.0005



# **Partie 3 :**

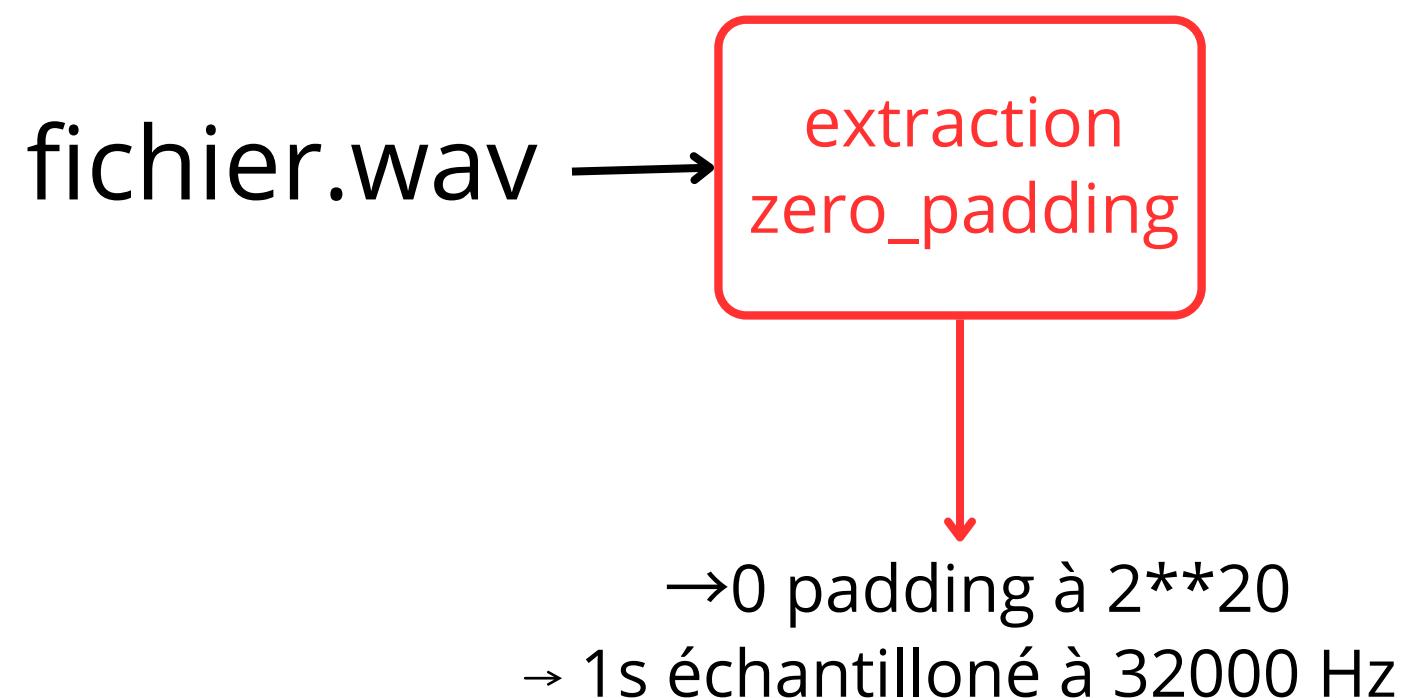
# **Apprentissage supervisé**

# Classification

- 2 fichiers : SONS et SONS-VC composés de 3000 et 1500 chants de 3 oiseaux différents
- **Objectif** : classifier automatiquement un .wav selon l'oiseau

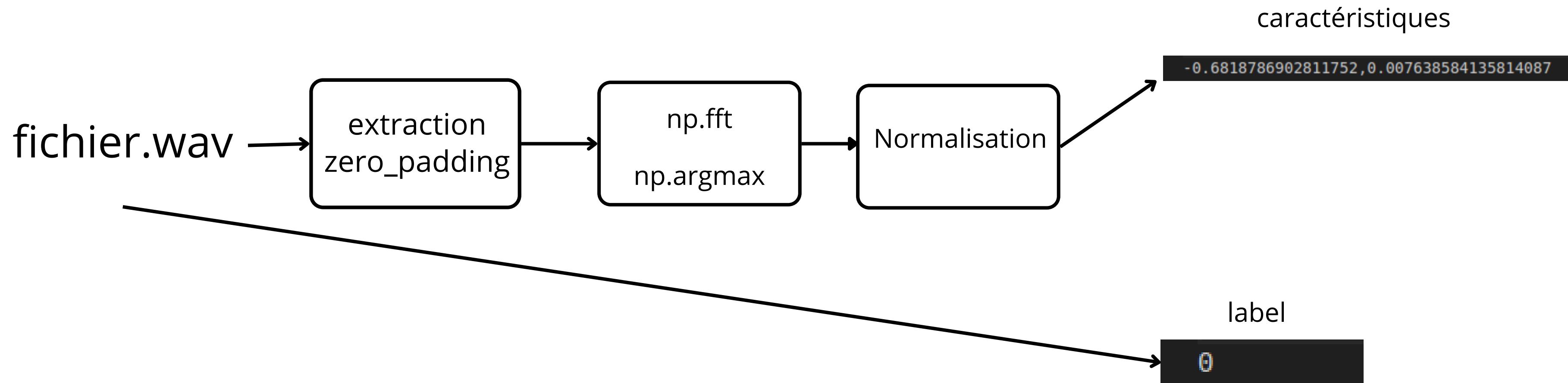
# Classification

--> Traitement des données :



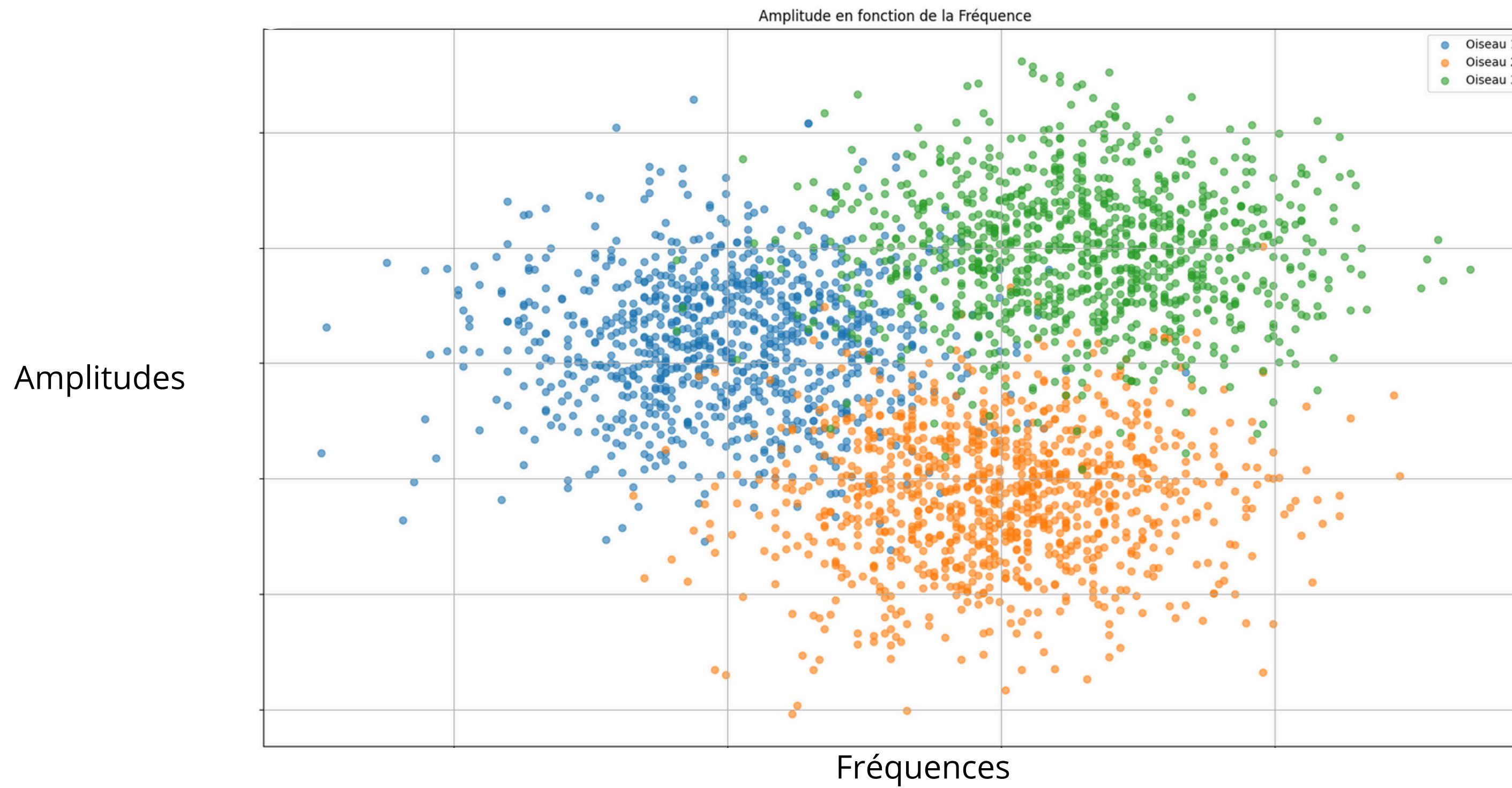
# Classification

--> Traitement des données :



# Classification

→ Traitement des données :



# Classification

--> Comment faire ? :

caractéristiques

```
1 0.1  
2 -0.722533243823236, 0.011951037431680626  
3 -1.0055645526770494, 1.2931362891780191  
4 -0.9206551600299408, -0.7149934527985522  
5 2.392417966060693, -0.5324152444124152  
6 -1.3452021233016125, 0.7033508027197481  
7 -1.0055645526770494, 0.9024975585803169  
8 -0.49610819674023693, -0.37075981341127195  
9 -1.1187770762185811, -0.6642765045630724  
10 -0.1564706261156416, 0.2834033815664998  
11 2.024477264550803, -0.49454456231344895  
12 -1.7697490865823486, 0.2357200979151866  
13 -2.1093866572069118, -0.6106545220368486  
14 1.175383337989331, 0.254474730019699  
15 -1.6848396939262078, -0.004373647325916855  
16 -0.8357457673647678, 0.9378730514229279
```

kernel gaussien

--> On définit une fonction phi ainsi :

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \quad \exp(-\gamma \|x - x'\|^2),$$

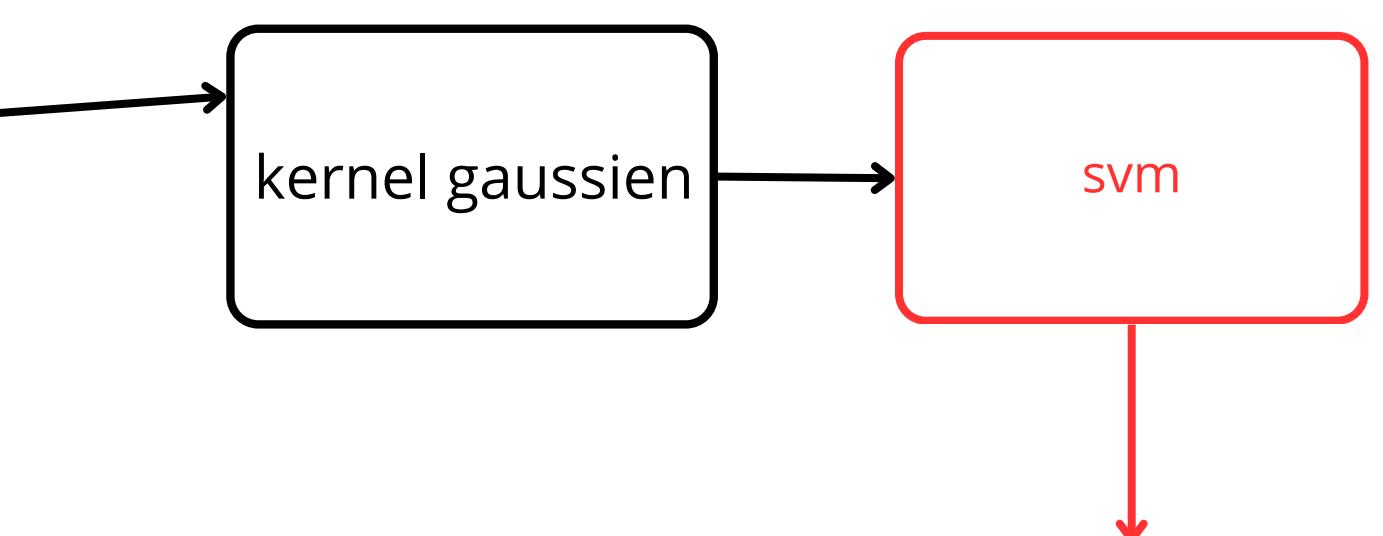
$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right)$$

# Classification

--> Comment faire ? :

caractéristiques

```
1 0,+
2 -0.722533243823236, 0.011951037431680626
3 -1.0055645526770494, 1.2931362891780191
4 -0.9206551600209408, -0.7149934527985522
5 -2.392417966066093, -0.5324152444124152
6 -1.3452021233016125, 0.7033508027197481
7 -1.0055645526770494, 0.9024975585803169
8 -0.49610819674023693, -0.37075981341127195
9 -1.1187770762185811, -0.6642765045630724
10 -0.1564706261156416, 0.2834033815664998
11 -2.024477264550803, -0.49454456231344895
12 -1.7697490865823486, 0.2357200979151866
13 -2.1093866572069118, -0.6106545220368486
14 -1.175383337989331, 0.2544747300199699
15 -1.6848396939262078, -0.004373647325916855
16 -0.8357457673647678, 0.9378730514229279
```



$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^n \zeta_i$$

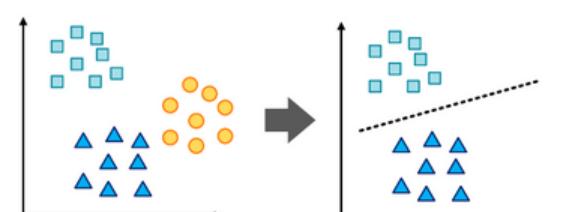
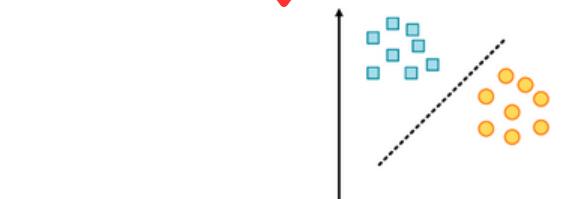
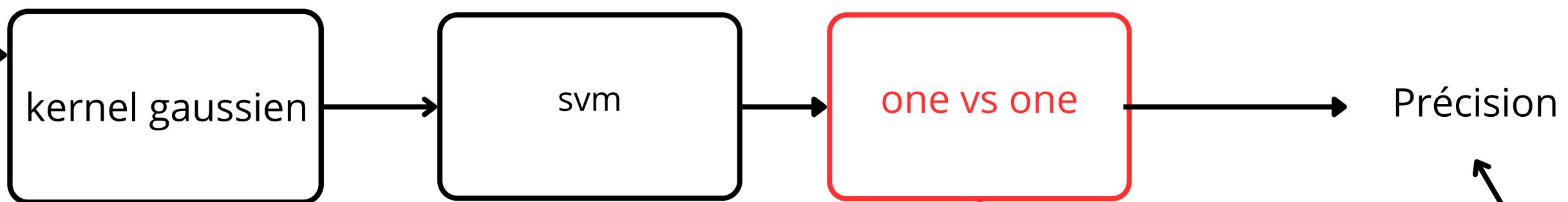
subject to  $y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$   
 $\zeta_i \geq 0, i = 1, \dots, n$

# Classification

--> Comment faire ? :

caractéristiques

```
2 -0.722533243823236, 0.811951037431680626
3 -1.0055645526770494, 1.2931362891780191
4 -0.9206551600209408, -0.7149934527985522
5 -2.392417966060693, -0.5324152444124152
6 -1.3452021233016125, 0.7033508027197481
7 -1.0055645526770494, 0.9024975585803169
8 -0.49610819674023693, -0.37075981341127195
9 -1.1187770762185811, -0.6642765045630724
10 -0.1564706261156416, 0.2834033815664998
11 -2.024477264550803, -0.49454456231344895
12 -1.7697498865823486, 0.2357200979151866
13 -2.1093866572069118, -0.6106545220368486
14 -1.175383337989331, 0.2544747300199699
15 -1.6848396939262078, -0.004373647325916855
16 -0.8357457673647678, 0.9378730514229279
```



```
1 Label
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 0
```

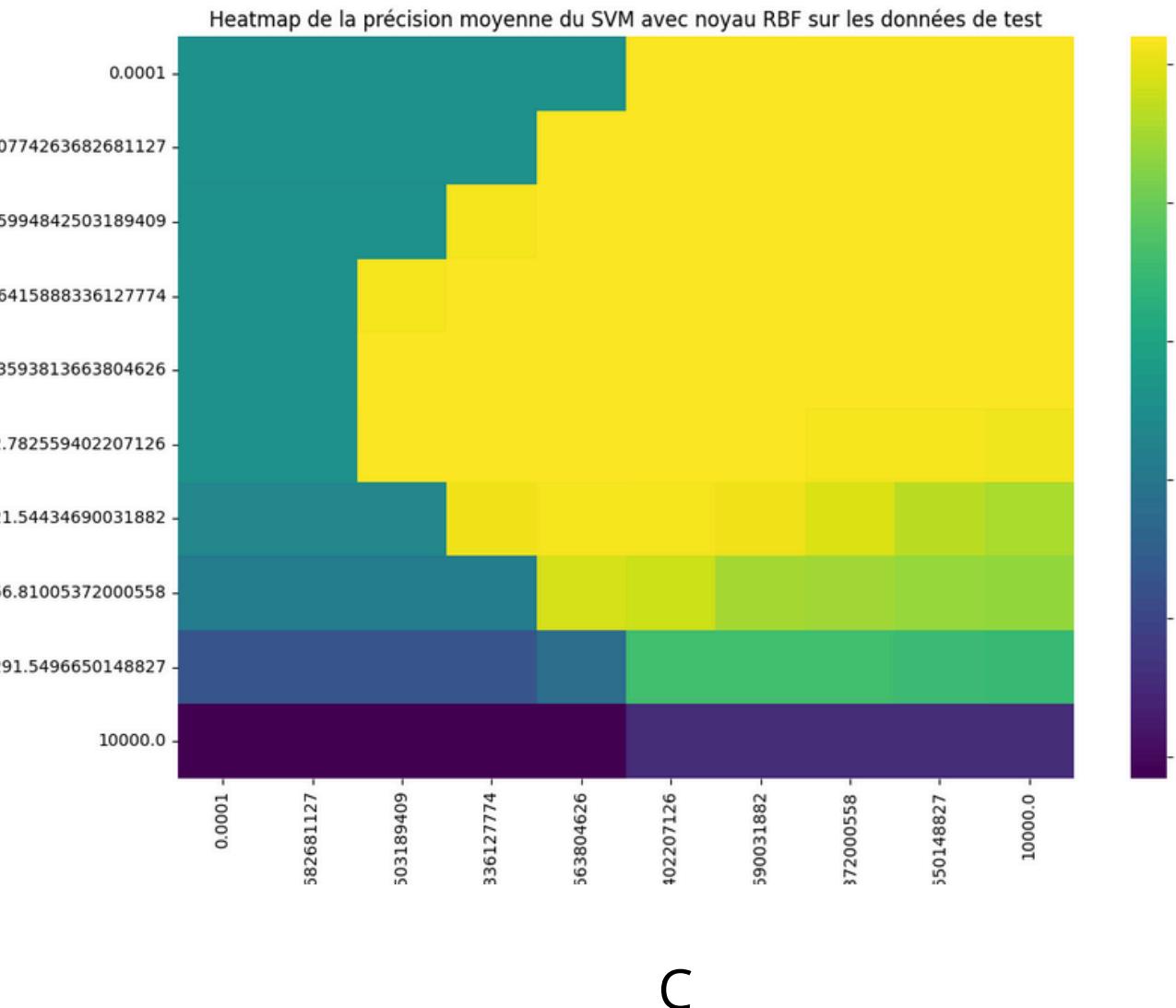
labels

# Classification

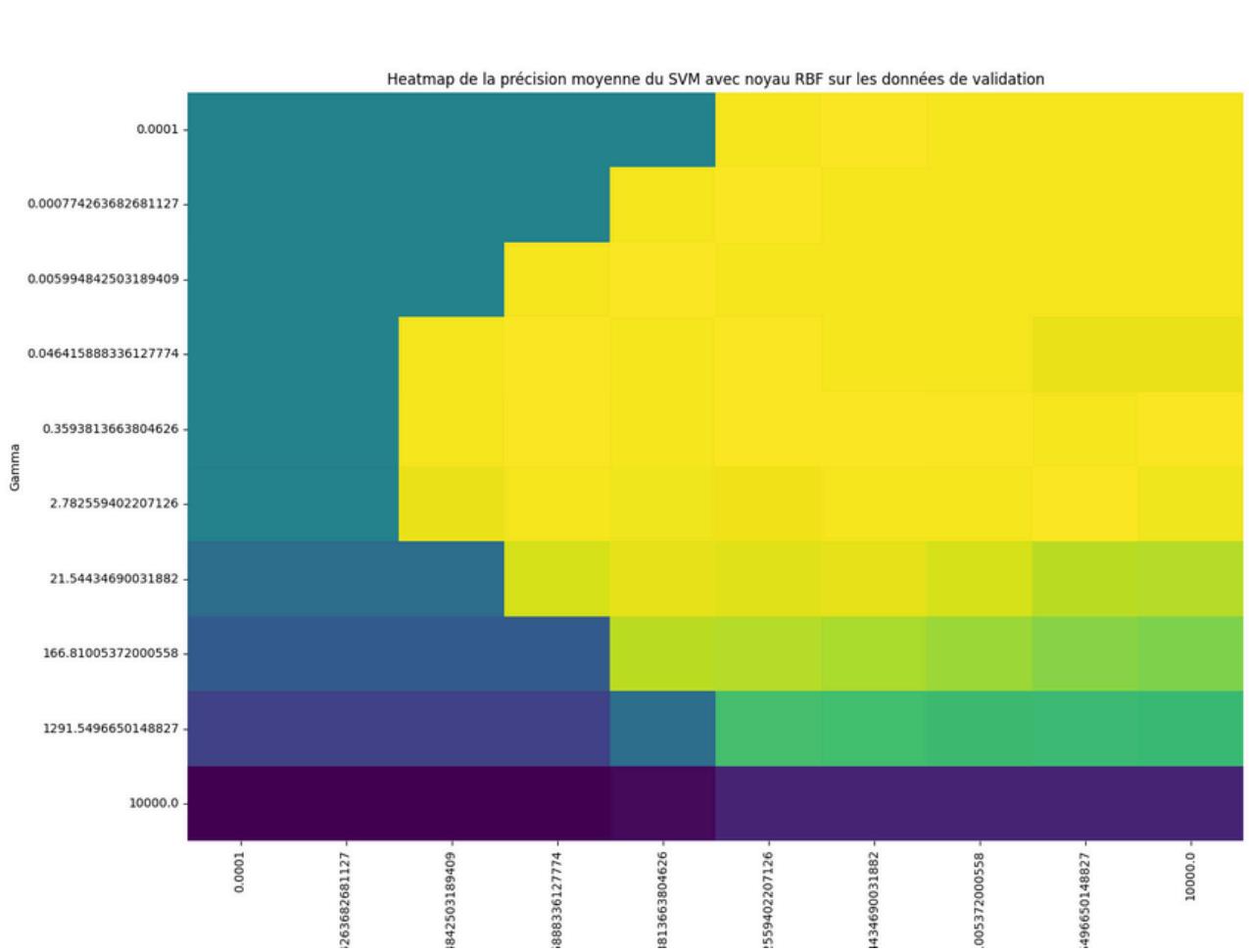
## → RESULTATS :

### APPRENTISSAGE

GAMMA



SONS-VC



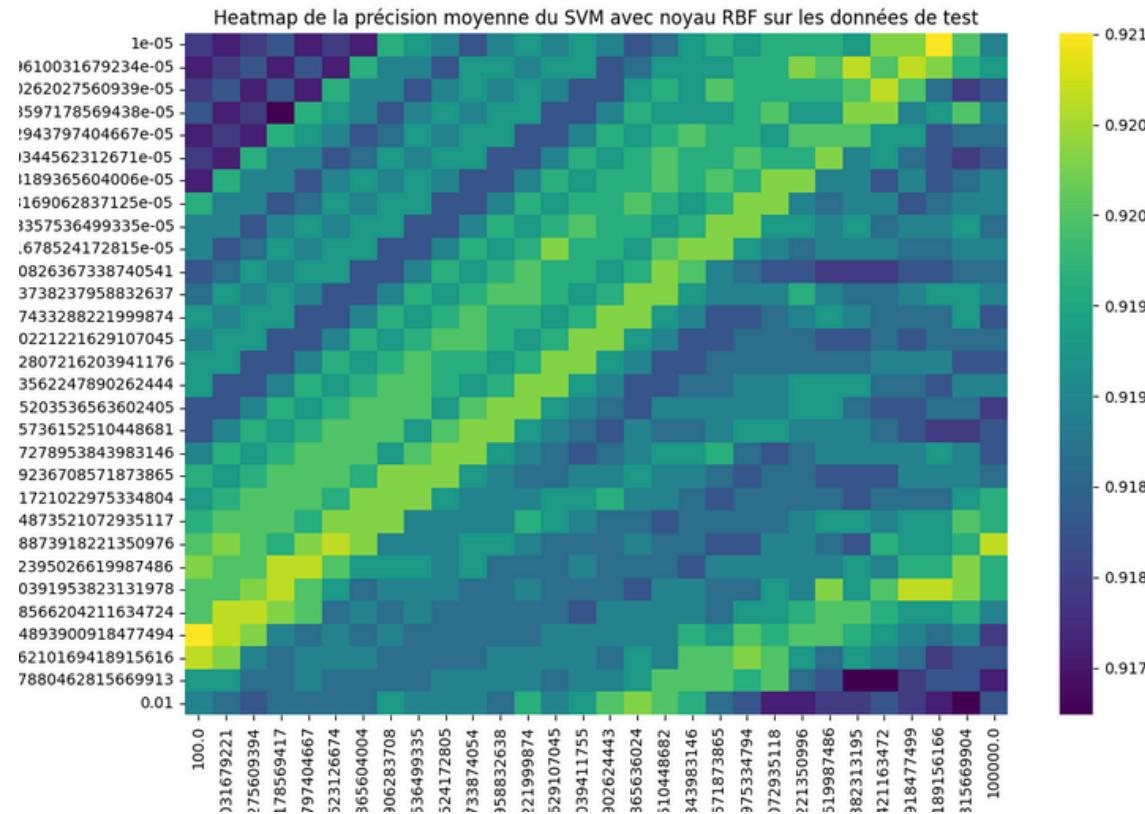
→ Intuitivement, on peut se dire que le meilleur résultat sera atteint pour C très élevé et gamma très faible

# Classification

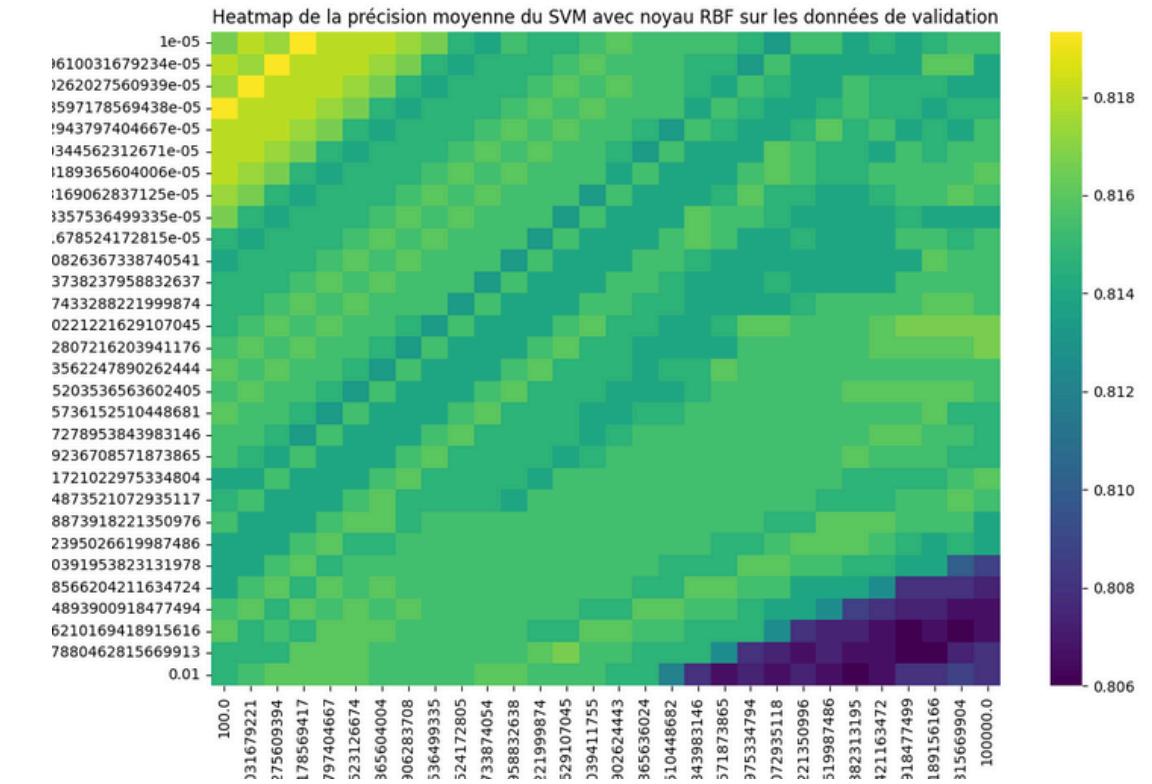
## → RESULTATS :

APPRENTISSAGE

SONS - VC



C



C

GAMMA

--> On recommence avec des valeurs de C et Gamma différentes, cette fois ci moins facile de cerner le pattern. on voit quand même que en haut à gauche de SONS- VC on peut choisir de meilleurs paramètres

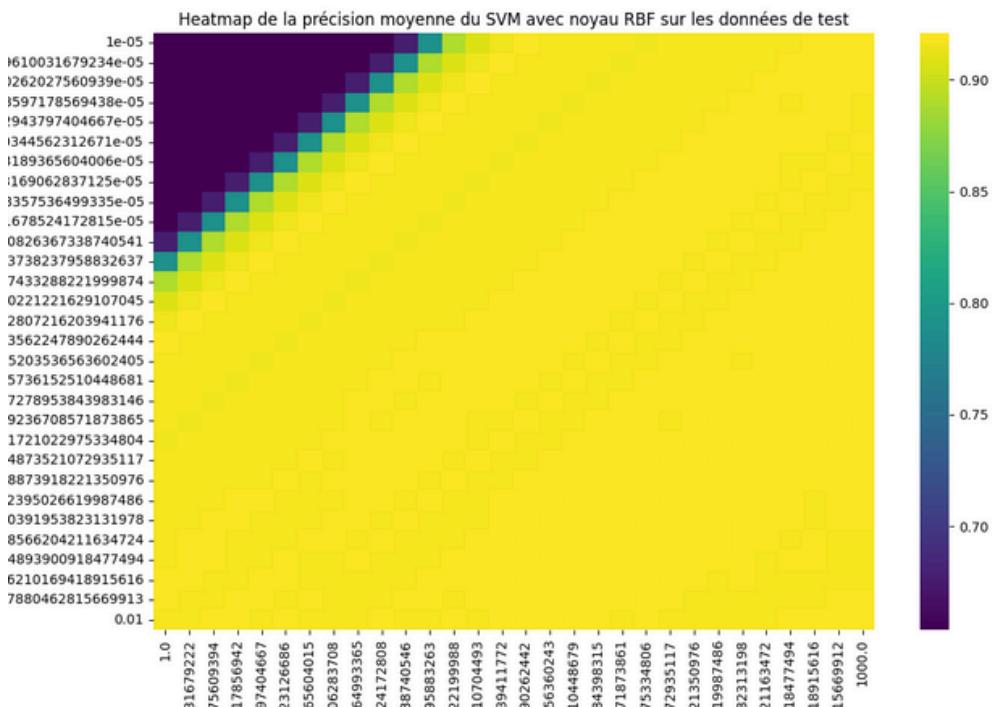
# Classification

## → RESULTATS :

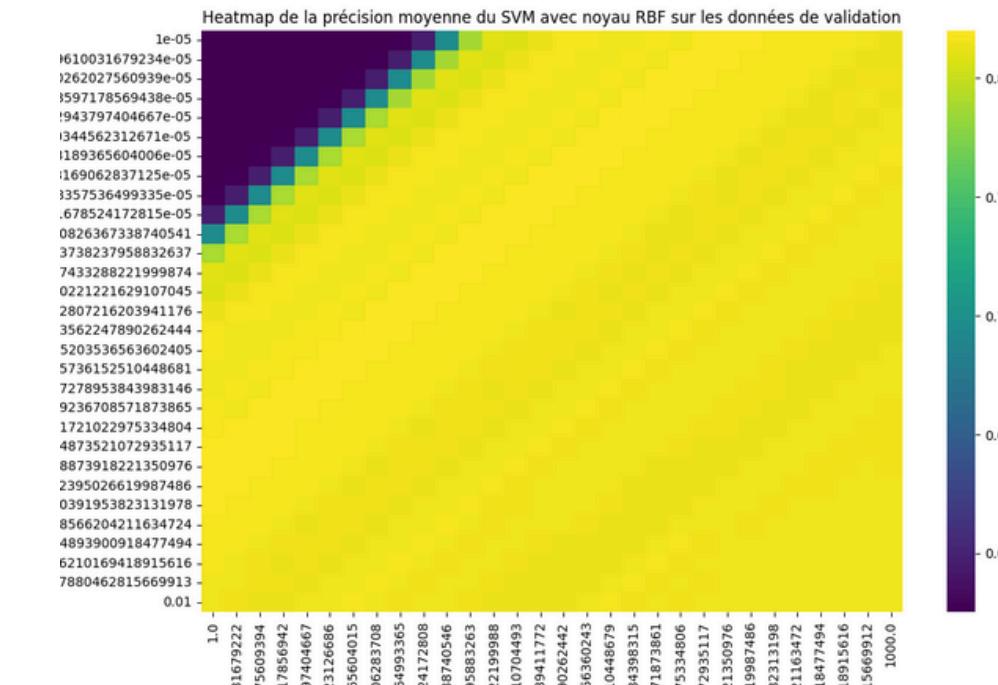
APPRENTISSAGE

SONS - VC

GAMMA



C



C

# Classification

→ RESULTATS :

**ENTRAINEMENT SUR “SONS, TEST SUR “SONS”**

```
gamma = 0.01 C = 45.20353656360243 et a une valeur de 0.9210039807985012
```

**ENTRAINEMENT SUR “SONS, TEST SUR “SONS-VC”**

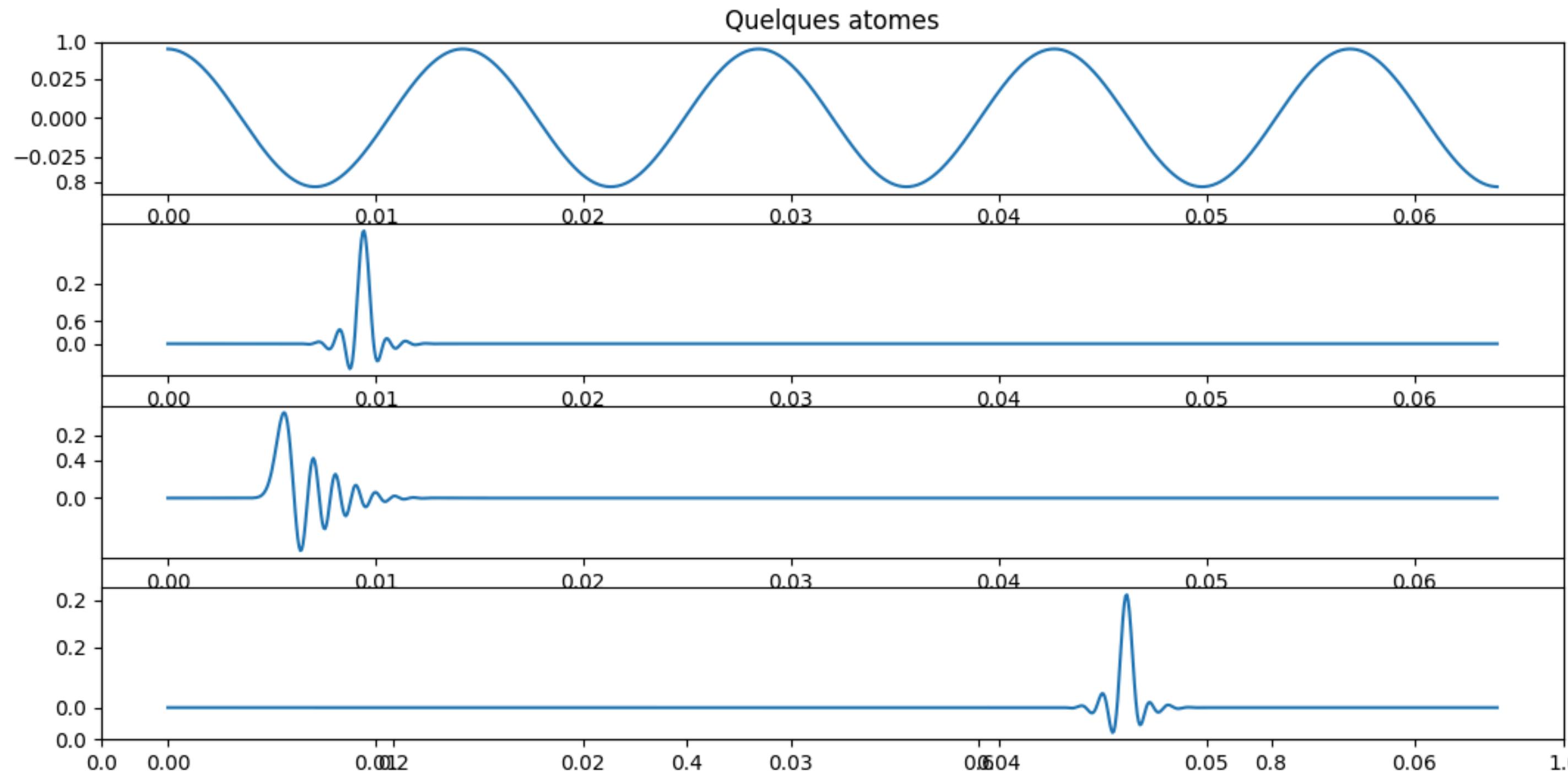
```
gamma = 5.2983169062837125e-05C = 45.20353656360243 et a une valeur de 0.82
```

# **Partie 4 :**

# **Compression de signaux**

- réimplémentation python (temps plus long)**
- signal vocal de ~1s**
- échantillonage à 16kHz**
- fenêtres de longueur 1024 espacées de 512**

# Création du dictionnaire

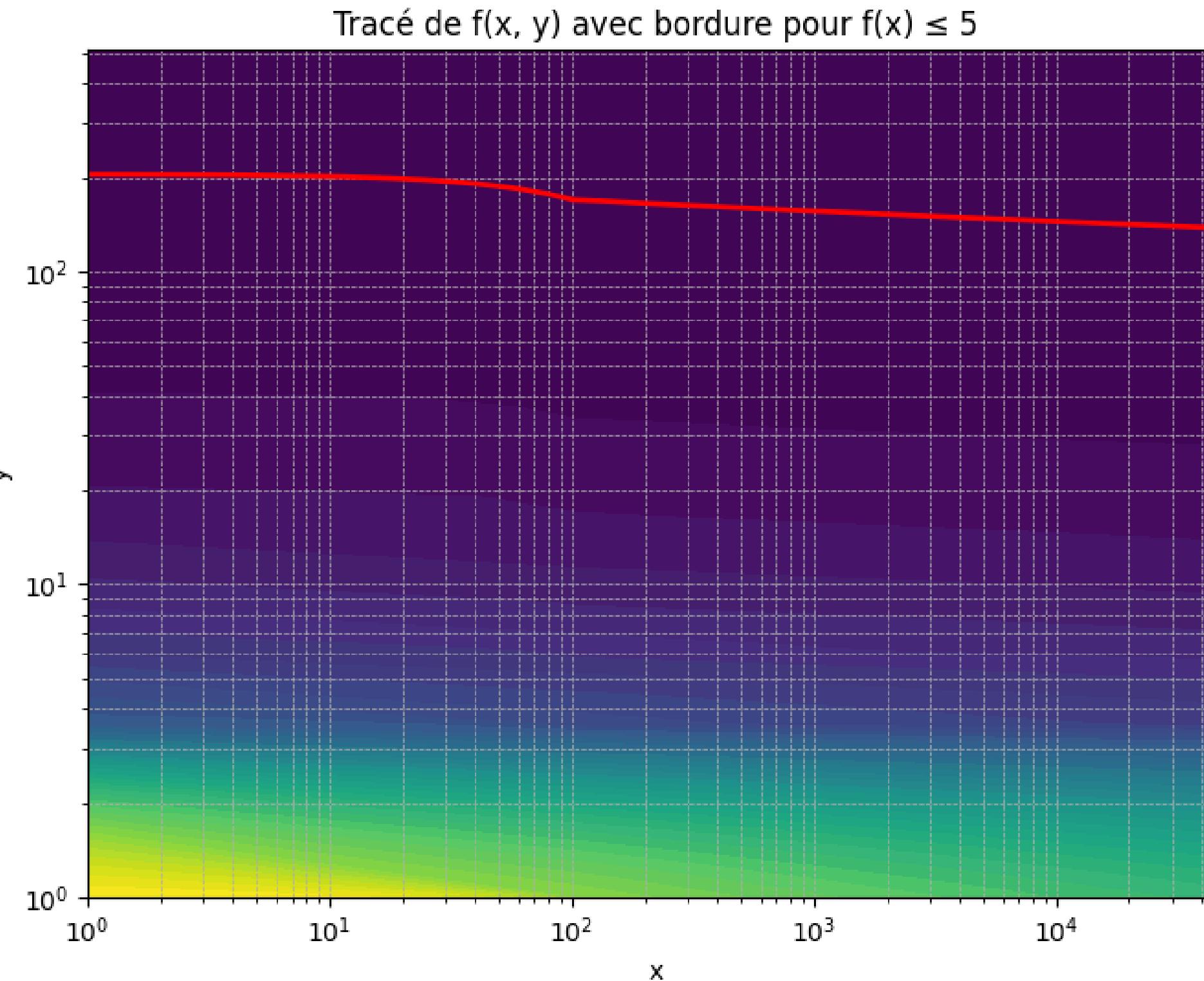


On prend:

- DCT
- ondelettes mères et fillés décalées en temporel

A cause du produit scalaire, fourrier et DCT sont redondants

# Dimensionnement du dictionnaire



$$C = \frac{N \times b_{\text{signal}}}{K \times (b_{\text{index}} + b_{\text{coeff}})}$$

On trace avec  $K=y$  et  $bc=\log_2(x)$

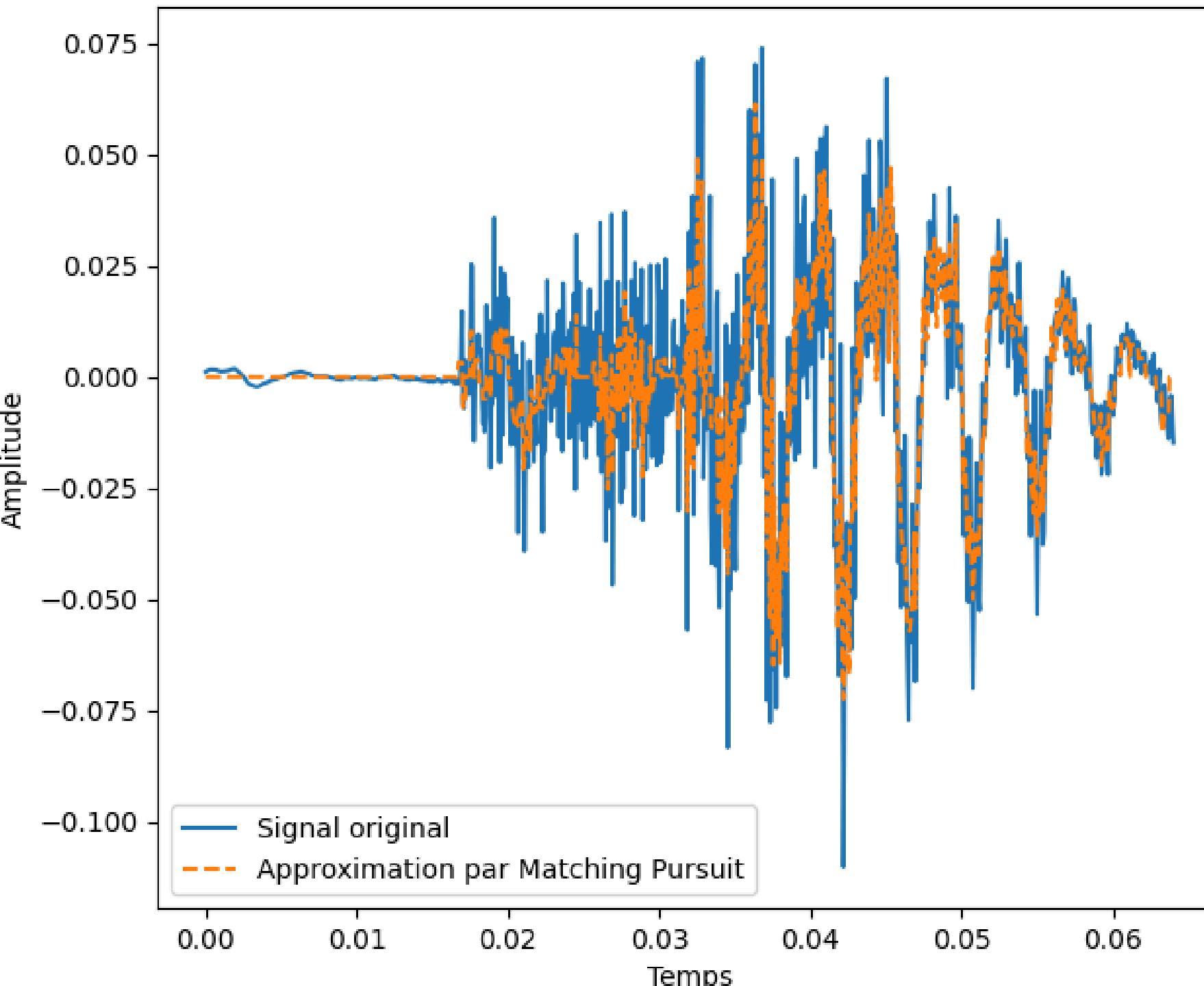
Il faut moins de  $\text{maxit} \leq 200$  pour garantir une bonne compression

Vous aviez mentionné de  $C \sim 50$  mais pour quel maxit/taille de dictionnaire?

**Le temps de calcul est long déjà pour 4000 atomes!**

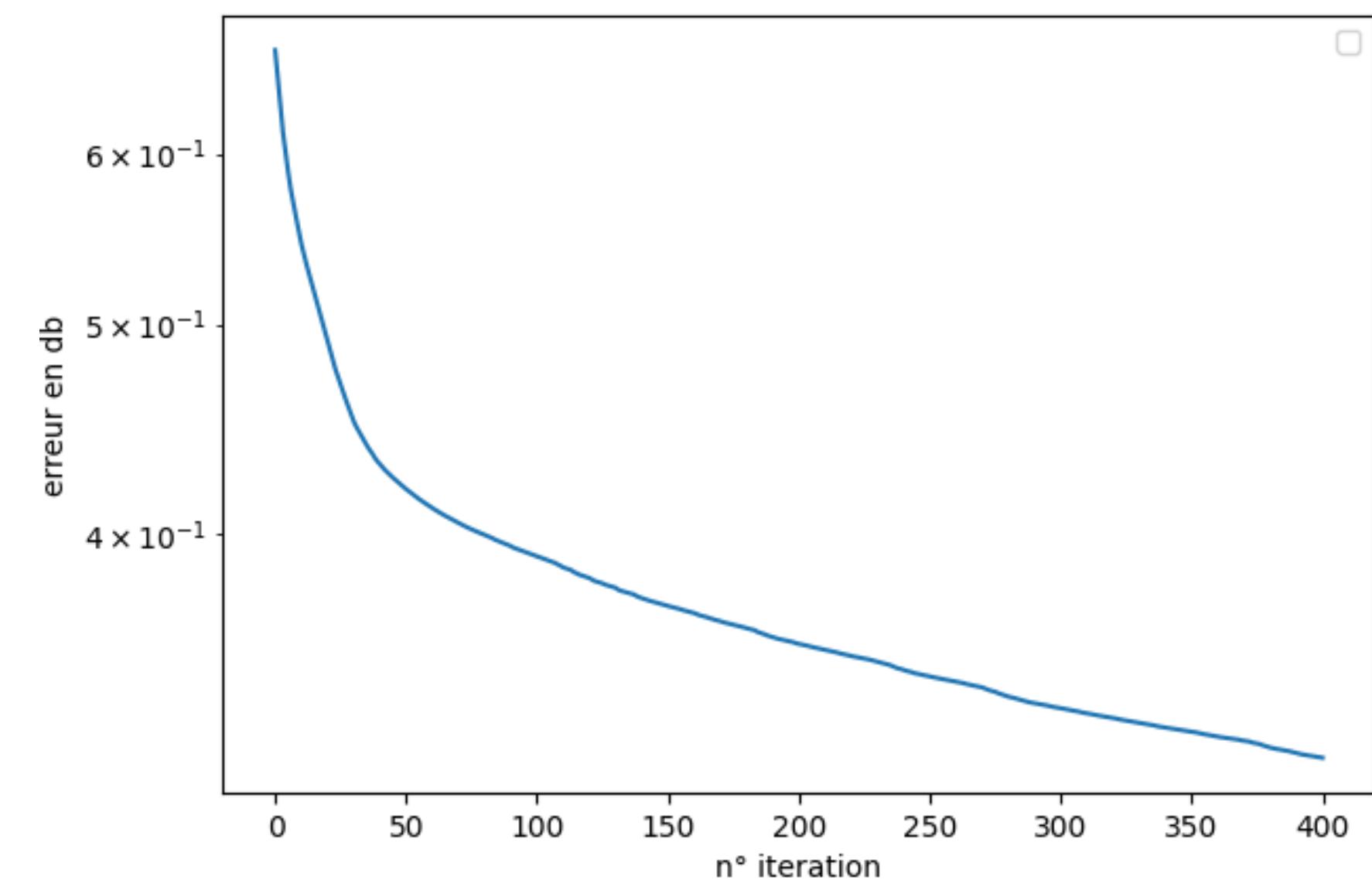
# Pursuit - Sur une fenêtre

Approximation du signal par Matching Pursuit avec ondelettes



max\_it = 400

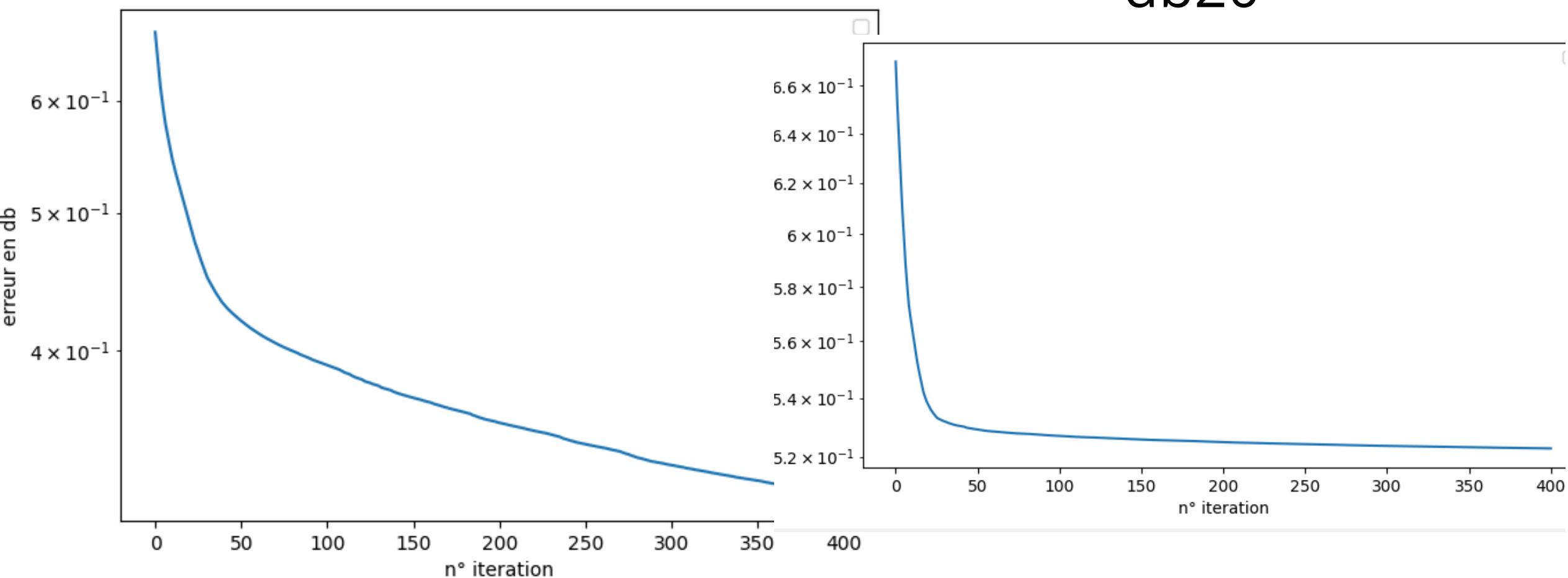
erreur en dB en fonction de l'itération



Dictionnaire avec les ondelettes  
filles et meres avec différents  
déphasages de sym20 et db1

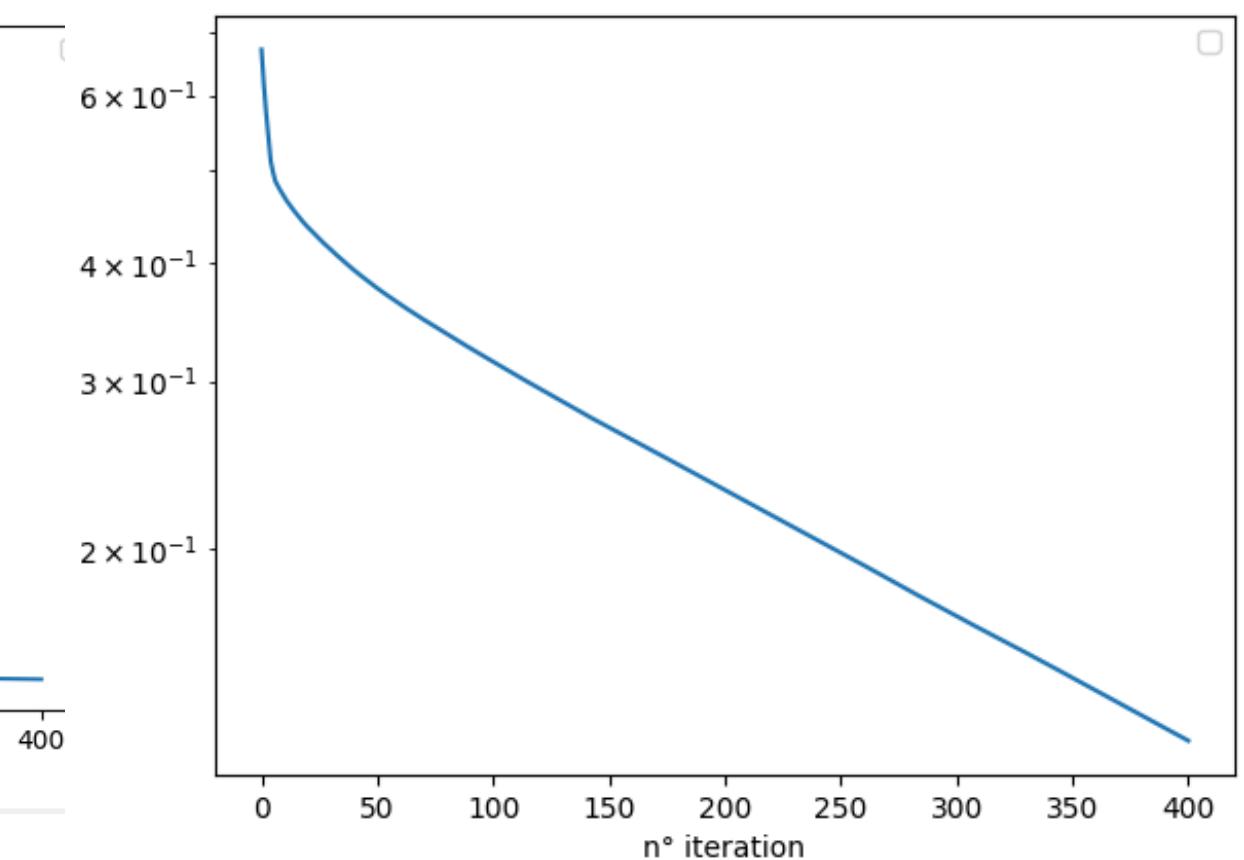
# Matching pursuit - Différents dictionnaires

sym20 et db1



db20

haar et DCT



*Le choix du dictionnaire importe!*

il faut être parcimonieux pour:

- taille du dictionnaire (Temps de calcul)
- les atomes choisis (cohérence avec les signal)
- max\_it (taux de compression)



Par itération on trouve **DCT**,  
**rbio3.3, bior3.5, bior5.5, bior6.8**  
comme meilleur compromis

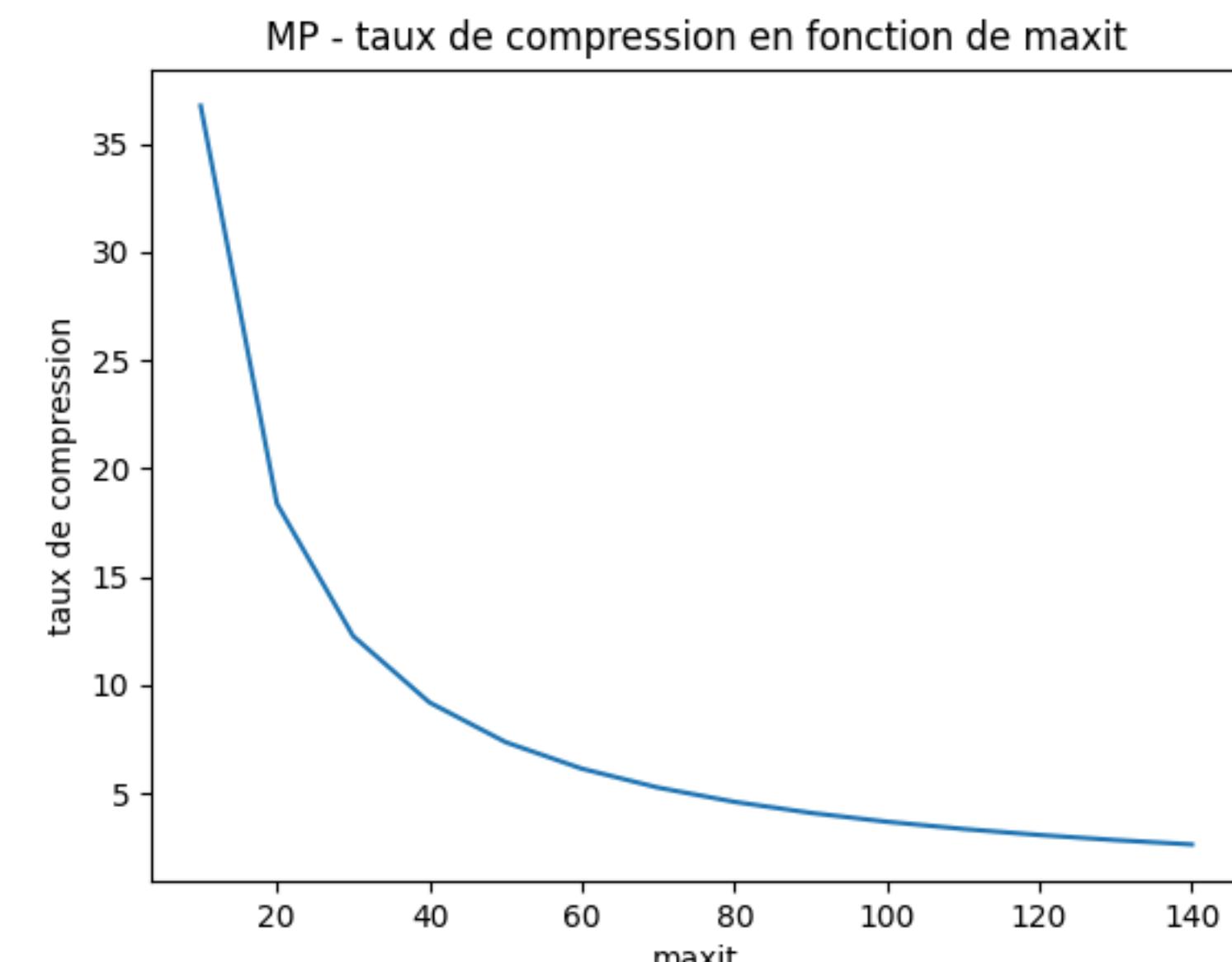
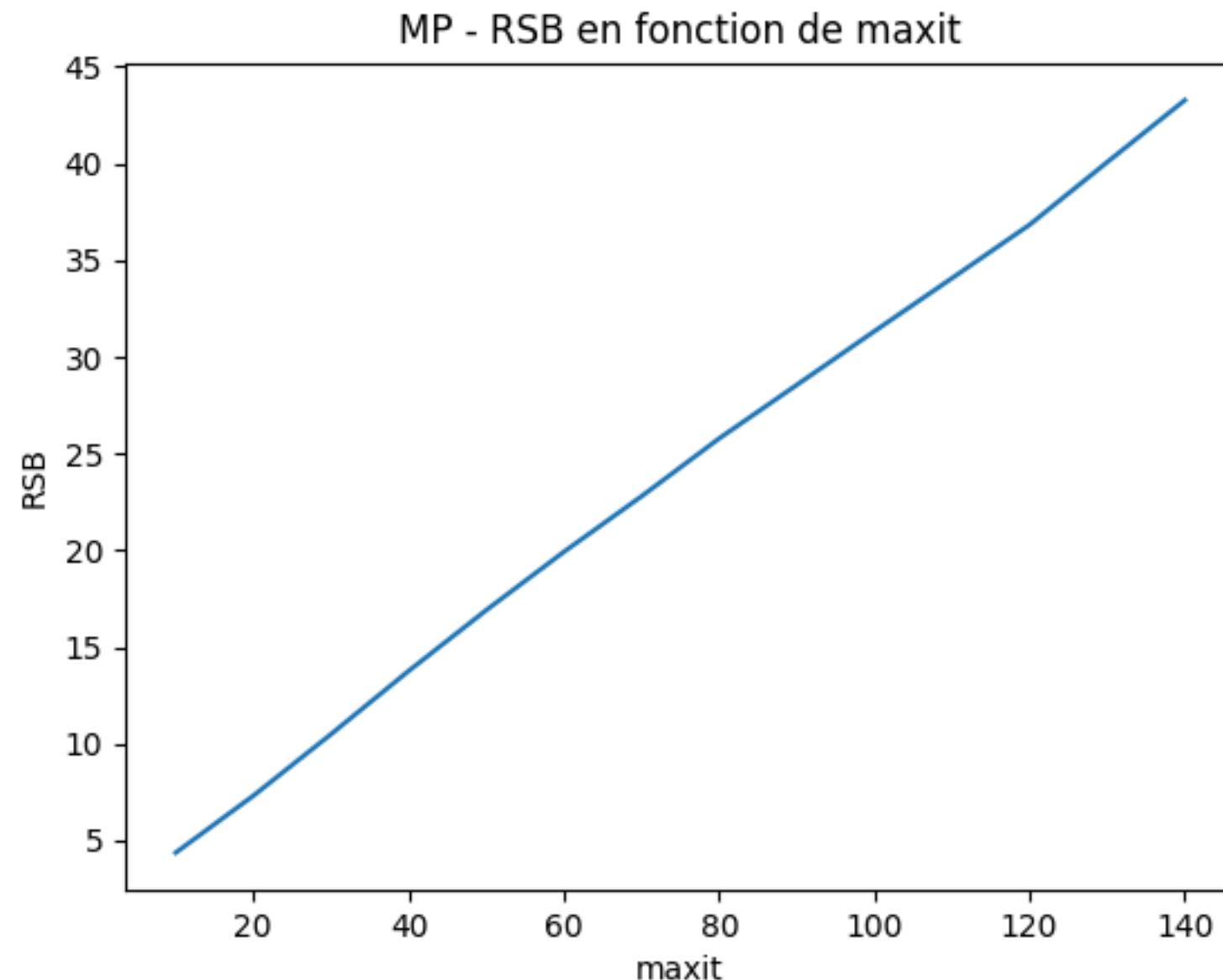
# Pursuit - Sur un signal entier



**Rectangle : Problèmes au bords**

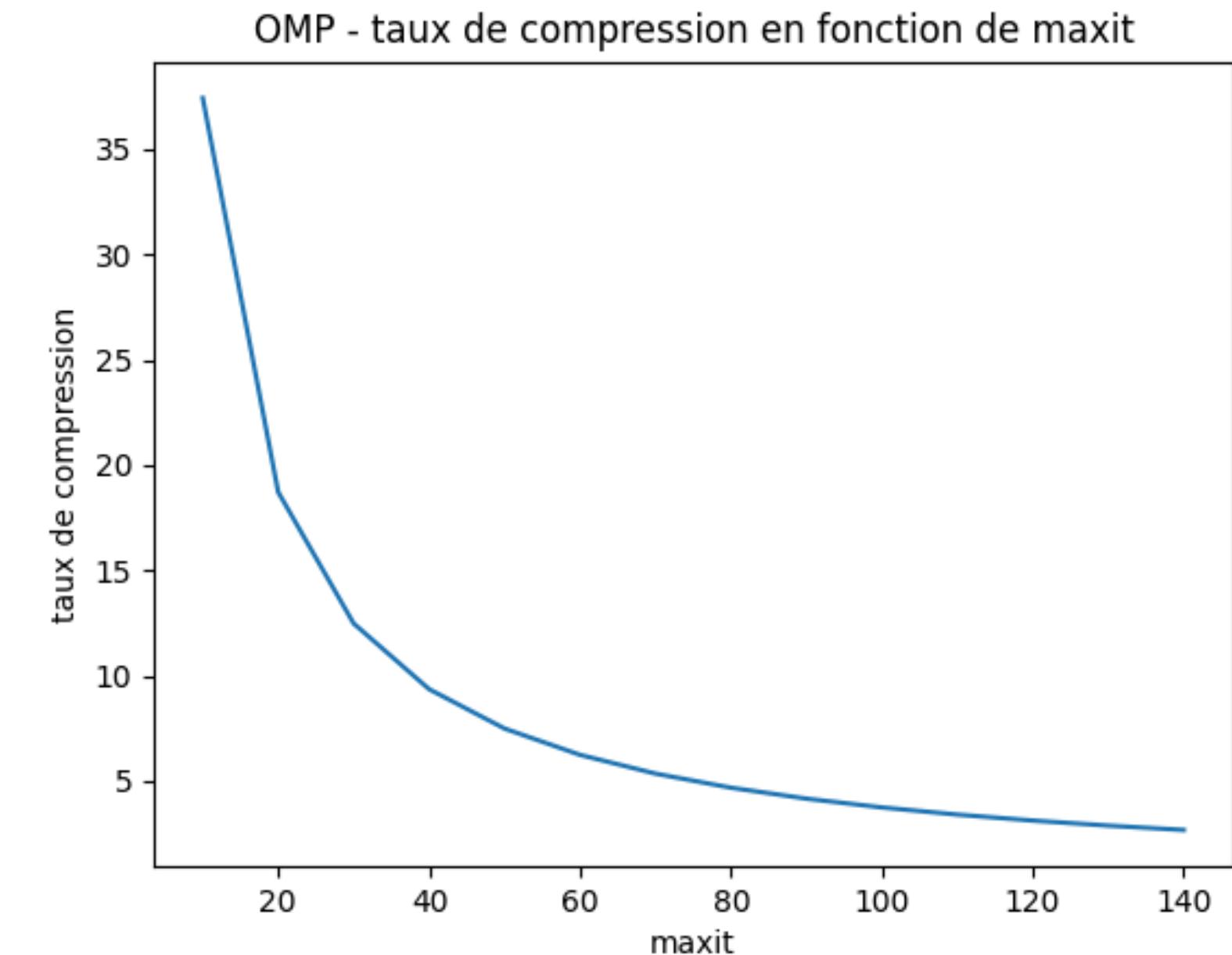
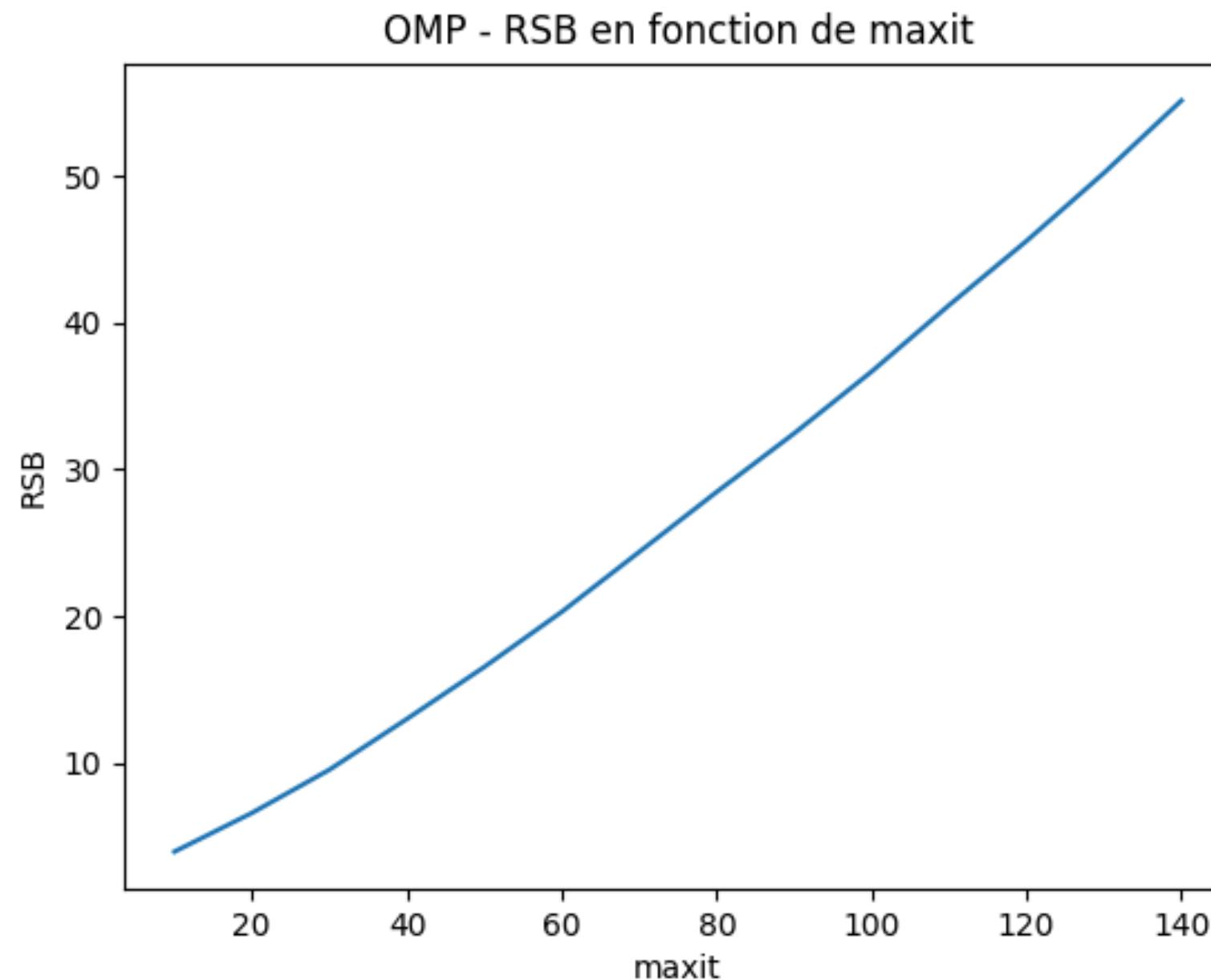
**hanning, hamming, triangle : Résultats similaires**

# Matching pursuit



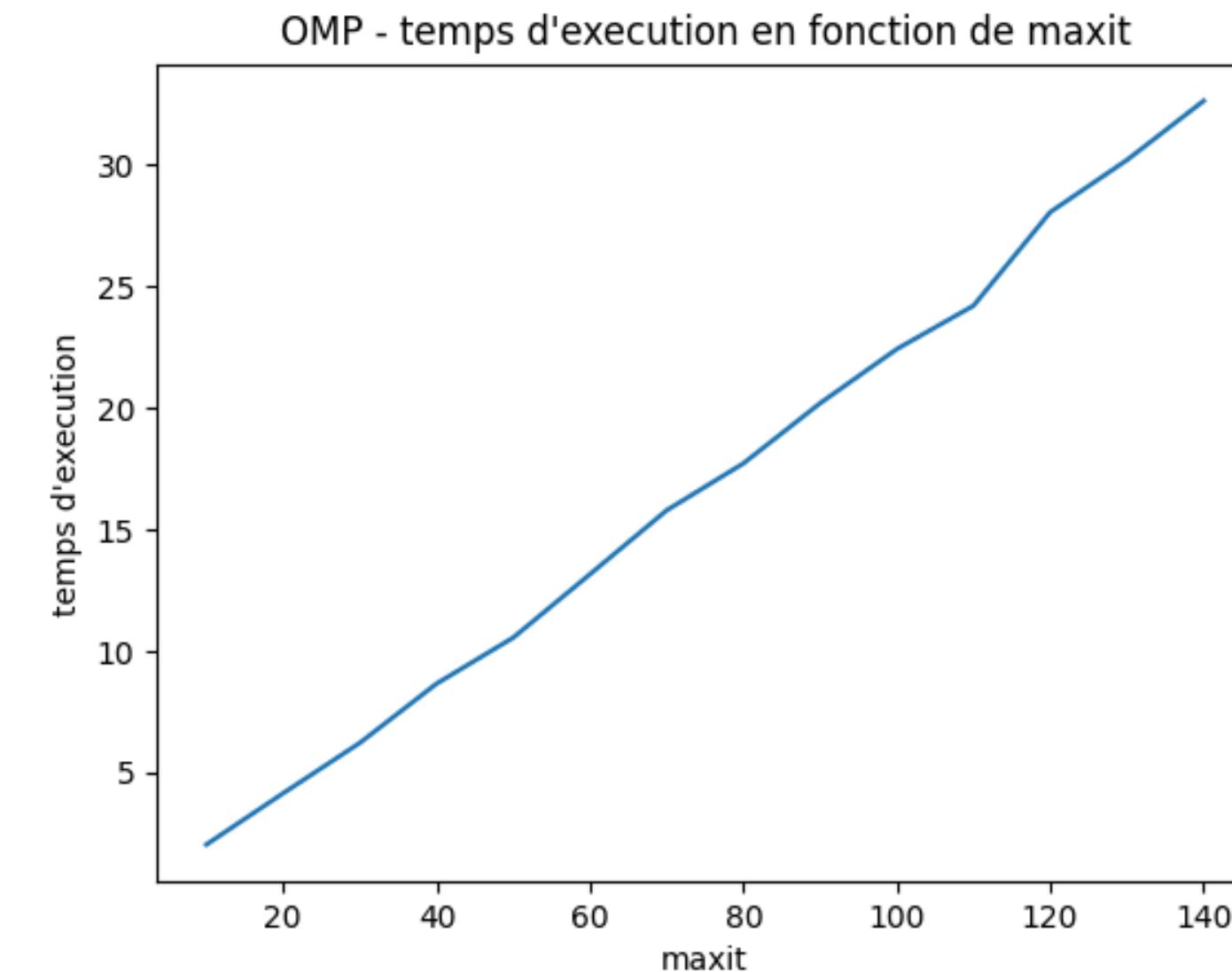
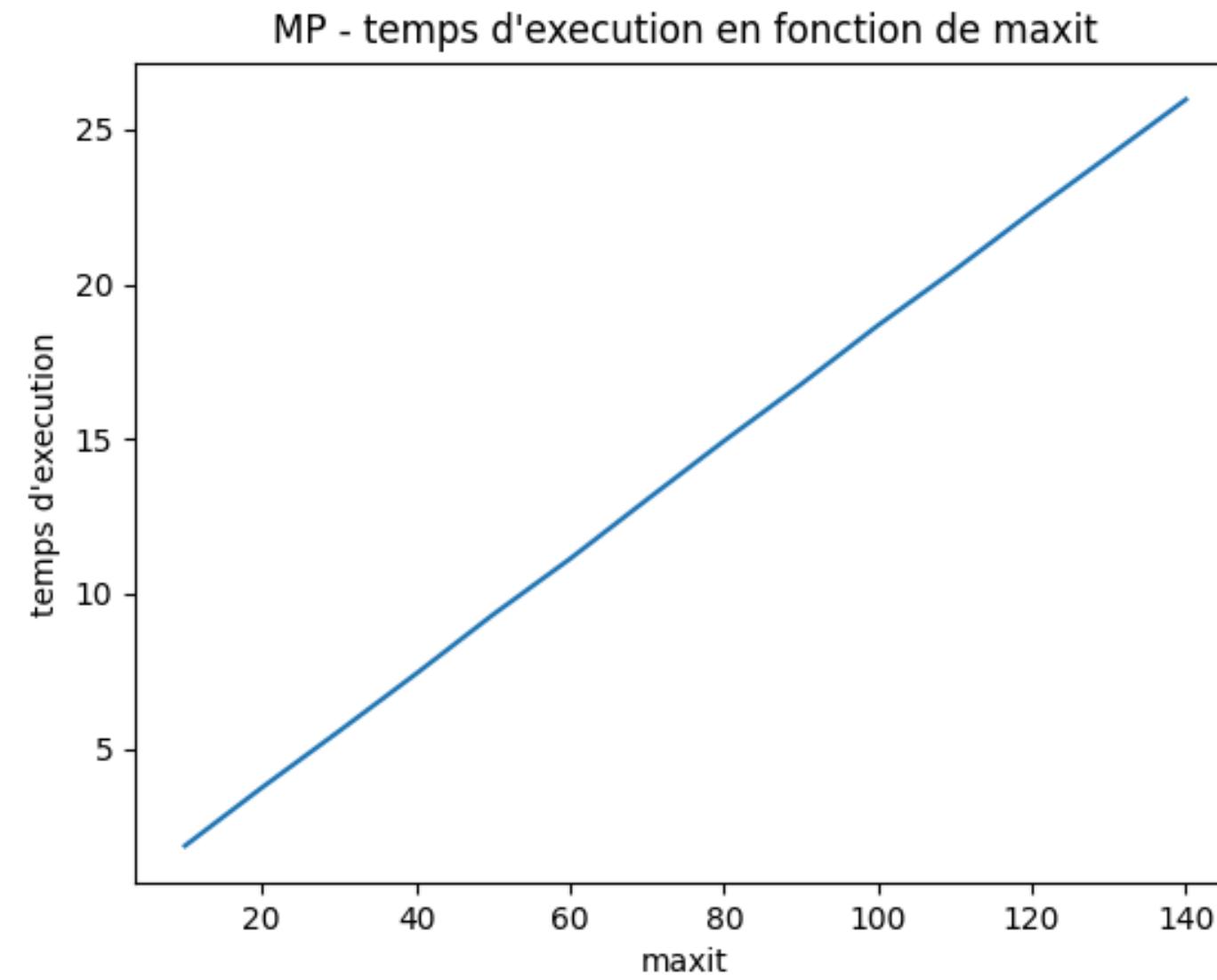
il faut **RSB>30** pour une bonne reconstruction soit **maxit=80 et tcomp=5**

# Orthogonal matching pursuit



Les courbes sont similaires (RSB un peu meilleur)

# Comparaison des méthodes



**OMP beaucoup plus long pour maxit élevé**

**Similaire autour de maxit = 80**

**OMP ne vaut pas le coup en temps de calcul !**

# Basis Pursuit

**Pour basis pursuit il faut  $\mathbf{Ax} = \mathbf{b}$**   
**Dictionnaire de petite taille  $\mathbf{x}$  n'existe pas**  
**L'algorithme ne converge pas**

**Merci de votre attention !**